DEPARTAMENTO DE LENGUAJES Y SISTEMAS INFORMÁTICOS E INGENIERÍA DEL SOFTWARE

Facultad de Informática
Universidad Politécnica de Madrid

TESIS DOCTORAL

Modelo de Arquitectura para Gestión Cooperativa de Sistemas y Servicios Distribuidos basado en Agentes Autónomos

Autor
Francisco Javier Soriano Camino
Licenciado en Informática

Directores
Fernando Alonso Amo
Doctor en Informática

Genoveva López Gómez
Doctora en Informática

Año: 2003

A mi mujer Ana Belén,
a mis padres Luciano y María,
y a mi hermana Pilar.

Jamás se descubriría nada
si nos considerásemos satisfechos con las cosas descubiertas.
Séneca el Filósofo, Lucius Annaeus (c.5 a. C.-65 d. C.)

Agradecimientos

Esta sección debería nombrar a todas aquellas personas que, de una u otra forma, me han apoyado y animado a lo largo de estos últimos años, sin las cuales no hubiese sido posible la culminación con éxito de un proyecto de tanta magnitud como es la elaboración de una Tesis Doctoral. Sin embargo, soy consciente de que muchas de ellas no aparecen citadas. Sirvan estas líneas para hacerles llegar a todas ellas mi agradecimiento y mi consideración.

Una vez dicho esto, me gustaría hacer especial mención de mis directores de Tesis, los doctores D. Fernando Alonso Amo y Dña. Genoveva López Gómez, a quienes estimo y admiro. Esta Tesis no hubiese sido posible sin su estímulo, perseverancia y experiencia. Les estoy sinceramente agradecido por la confianza que siempre han depositado en mí.

Quisiera expresar también mi más profundo sentimiento de gratitud y devoción a mi familia, que ha sido siempre para mí fuente incondicional de amor, apoyo e inspiración; especialmente a mi mujer Ana Belén, a mis padres Luciano y María, y a mi hermana Pilar, a quienes va dedicado este trabajo. Ellos me han dado la oportunidad de realizarlo en la mejor de las condiciones posibles.

No quisiera finalizar estos agradecimientos sin mencionar a mis compañeros del Grupo de Redes de Computadores y Sistemas Distribuidos y de CETTICO. Esto incluye a José Luis, Javier, Nicolás, Carlos, Luis, Sonia, Daniel, Pepe, Löic, José María, Cesar y Agustín. Les aprecio y estoy agradecido por su amistad, y por haberme proporcionado un magnífico contexto en el que poder desarrollar mi actividad investigadora y, en general, mi vida profesional.

Resumen

La creciente complejidad, heterogeneidad y dinamismo inherente a las redes de telecomunicaciones, los sistemas distribuidos y los servicios avanzados de información y comunicación emergentes, así como el incremento de su criticidad e importancia estratégica, requieren la adopción de tecnologías cada vez más sofisticadas para su gestión, su coordinación y su integración por parte de los operadores de red, los proveedores de servicio y las empresas, como usuarios finales de los mismos, con el fin de garantizar niveles adecuados de funcionalidad, rendimiento y fiabilidad.

Las estrategias de gestión adoptadas tradicionalmente adolecen de seguir modelos excesivamente estáticos y centralizados, con un elevado componente de supervisión y difícilmente escalables. La acuciante necesidad por flexibilizar esta gestión y hacerla a la vez más escalable y robusta, ha provocado en los últimos años un considerable interés por desarrollar nuevos paradigmas basados en modelos jerárquicos y distribuidos, como evolución natural de los primeros modelos jerárquicos débilmente distribuidos que sucedieron al paradigma centralizado. Se crean así nuevos modelos como son los basados en Gestión por Delegación, en el paradigma de código móvil, en las tecnologías de objetos distribuidos y en los servicios web. Estas alternativas se han mostrado enormemente robustas, flexibles y escalables frente a las estrategias tradicionales de gestión, pero continúan sin resolver aún muchos problemas.

Las líneas actuales de investigación parten del hecho de que muchos problemas de robustez, escalabilidad y flexibilidad continúan sin ser resueltos por el paradigma jerárquico-distribuido, y abogan por la migración hacia un paradigma cooperativo fuertemente distribuido. Estas líneas tienen su germen en la Inteligencia Artificial Distribuida (DAI) y, más concretamente, en el paradigma de agentes autónomos y en los Sistemas Multi-agente (MAS). Todas ellas se perfilan en torno a un conjunto de objetivos que pueden resumirse en alcanzar un mayor grado de autonomía en la funcionalidad de la gestión y una mayor capacidad de autoconfiguración que resuelva los problemas de escalabilidad y la necesidad de supervisión presentes en los sistemas actuales, evolucionar hacia técnicas de control fuertemente distribuido y cooperativo guiado por la meta y dotar de una mayor riqueza semántica a los modelos de información. Cada vez más investigadores están empezando a utilizar agentes para la gestión de redes y sistemas distribuidos. Sin embargo, los límites establecidos en sus trabajos entre agentes móviles (que siguen el paradigma de código

móvil) y agentes autónomos (que realmente siguen el paradigma cooperativo) resultan difusos. Muchos de estos trabajos se centran en la utilización de agentes móviles, lo cual, al igual que ocurría con las técnicas de código móvil comentadas anteriormente, les permite dotar de un mayor componente dinámico al concepto tradicional de Gestión por Delegación. Con ello se consigue flexibilizar la gestión, distribuir la lógica de gestión cerca de los datos y distribuir el control. Sin embargo se permanece en el paradigma jerárquico distribuido. Si bien continúa sin definirse aún una arquitectura de gestión fiel al paradigma cooperativo fuertemente distribuido, estas líneas de investigación han puesto de manifiesto serios problemas de adecuación en los modelos de información, comunicación y organizativo de las arquitecturas de gestión existentes.

En este contexto, la tesis presenta un modelo de arquitectura para gestión holónica de sistemas y servicios distribuidos mediante sociedades de agentes autónomos, cuyos objetivos fundamentales son el incremento del grado de automatización asociado a las tareas de gestión, el aumento de la escalabilidad de las soluciones de gestión, soporte para delegación tanto por dominios como por macro-tareas, y un alto grado de interoperabilidad en entornos abiertos. A partir de estos objetivos se ha desarrollado un modelo de información formal de tipo semántico, basado en lógica descriptiva que permite un mayor grado de automatización en la gestión en base a la utilización de agentes autónomos racionales, capaces de razonar, inferir e integrar de forma dinámica conocimiento y servicios conceptualizados mediante el modelo CIM y formalizados a nivel semántico mediante lógica descriptiva. El modelo de información incluye además un "mapping" a nivel de meta-modelo de CIM al lenguaje de especificación de ontologías OWL, que supone un significativo avance en el área de la representación y el intercambio basado en XML de modelos y meta-información. A nivel de interacción, el modelo aporta un lenguaje de especificación formal de conversaciones entre agentes basado en la teoría de actos ilocucionales y aporta una semántica operacional para dicho lenguaje que facilita la labor de verificación de propiedades formales asociadas al protocolo de interacción. Se ha desarrollado también un modelo de organización holónico y orientado a roles cuyas principales características están alineadas con las demandadas por los servicios distribuidos emergentes e incluyen la ausencia de control central, capacidades de reestructuración dinámica, capacidades de cooperación, y facilidades de adaptación a diferentes culturas organizativas. El modelo incluye un submodelo normativo adecuado al carácter autónomo de los holones de gestión y basado en las lógicas modales deontológica y de acción.

Abstract

The growing complexity, heterogeneity and dynamism inherent in telecommunications networks, distributed systems and the emerging advanced information and communication services, as well as their increased criticality and strategic importance, calls for the adoption of increasingly more sophisticated technologies for their management, coordination and integration by network operators, service providers and end-user companies to assure adequate levels of functionality, performance and reliability.

The management strategies adopted traditionally follow models that are too static and centralised, have a high supervision component and are difficult to scale. The pressing need to flexibilise management and, at the same time, make it more scalable and robust recently led to a lot of interest in developing new paradigms based on hierarchical and distributed models, as a natural evolution from the first weakly distributed hierarchical models that succeeded the centralised paradigm. Thus new models based on management by delegation, the mobile code paradigm, distributed objects and web services came into being. These alternatives have turned out to be enormously robust, flexible and scalable as compared with the traditional management strategies. However, many problems still remain to be solved.

Current research lines assume that the distributed hierarchical paradigm has as yet failed to solve many of the problems related to robustness, scalability and flexibility and advocate migration towards a strongly distributed cooperative paradigm. These lines of research were spawned by Distributed Artificial Intelligence (DAI) and, specifically, the autonomous agent paradigm and Multi-Agent Systems (MAS). They all revolve around a series of objectives, which can be summarised as achieving greater management functionality autonomy and a greater self-configuration capability, which solves the problems of scalability and the need for supervision that plague current systems, evolving towards strongly distributed and goal-driven cooperative control techniques and semantically enhancing information models. More and more researchers are starting to use agents for network and distributed systems management. However, the boundaries established in their work between mobile agents (that follow the mobile code paradigm) and autonomous agents (that really follow the cooperative paradigm) are fuzzy. Many of these approximations focus on

the use of mobile agents, which, as was the case with the above-mentioned mobile code techniques, means that they can inject more dynamism into the traditional concept of management by delegation. Accordingly, they are able to flexibilise management, distribute management logic about data and distribute control. However, they remain within the distributed hierarchical paradigm. While a management architecture faithful to the strongly distributed cooperative paradigm has yet to be defined, these lines of research have revealed that the information, communication and organisation models of existing management architectures are far from adequate.

In this context, this dissertation presents an architectural model for the holonic management of distributed systems and services through autonomous agent societies. The main objectives of this model are to raise the level of management task automation, increase the scalability of management solutions, provide support for delegation by both domains and macro-tasks and achieve a high level of interoperability in open environments. Bearing in mind these objectives, a descriptive logic-based formal semantic information model has been developed, which increases management automation by using rational autonomous agents capable of reasoning, inferring and dynamically integrating knowledge and services conceptualised by means of the CIM model and formalised at the semantic level by means of descriptive logic. The information model also includes a mapping, at the CIM metamodel level, to the OWL ontology specification language, which amounts to a significant advance in the field of XML-based model and metainformation representation and exchange. At the interaction level, the model introduces a formal specification language (ACSL) of conversations between agents based on speech act theory and contributes an operational semantics for this language that eases the task of verifying formal properties associated with the interaction protocol. A role-oriented holonic organisational model has also been developed, whose main features meet the requirements demanded by emerging distributed services, including no centralised control, dynamic restructuring capabilities, cooperative skills and facilities for adaptation to different organisational cultures. The model includes a normative submodel adapted to management holon autonomy and based on the deontic and action modal logics.

Índice general

Índice de figuras iv

Índice de cuadros vii

Capítulo 1. Introducción 1

1.1. Introducción y motivaciones 1
1.2. Contribución 7
1.3. Estructura de la tesis 8

Capítulo 2. Estado de la cuestión 11

2.1. La arquitectura de gestión de Internet 12
2.2. La arquitectura de gestión de OSI 18
2.3. La arquitectura DMI del DMTF 26
2.4. La arquitectura WBEM del DMTF 28
2.5. OMA como arquitectura de gestión 31
2.6. Gestión basada en políticas PBN 36
2.7. Análisis de las arquitecturas existentes 38

Capítulo 3. Modelo de Información: Estructura del Conocimiento de Gestión 47

3.1. Alcance de la propuesta 49
3.2. Conceptualización CIM-OWL 51
3.3. "Mapping" del metaesquema CIM al metamodelo OWL 55
3.4. Adecuación del "Mapping" CIM-OWL a la Lógica Descriptiva 76
3.5. Servicios de razonamiento automático 91

Capítulo 4. Modelo de Información: Interfaz de interacción 95

4.1. Del modelado de actos hablados al modelado de conversaciones 97
4.2. Proceso de especificación de una interfaz DIA 99
4.3. Requisitos de un lenguaje de especificación de protocolos 104
4.4. El lenguaje de especificación ACSL 106

4.5. Verificación y evaluación semántica de especificaciones ACSL 121

Capítulo 5. Modelo de Organización **139**
5.1. Aspectos organizativos de la gestión 140
5.2. Necesidad de un nuevo modelo de organización de tipo cooperativo . 142
5.3. Organizaciones holónicas 144
5.4. Modelo de Organización Holónico 145
5.5. Caso de aplicación: Gestión holónica de los aspectos de seguridad de un SLA 154
5.6. Políticas sociales 158
5.7. Dinámica de las obligaciones 166
5.8. Creación, derogación y distribución de políticas mediante actos ilocucionales 168

Capítulo 6. Experimentación y Resultados **171**
6.1. Diseño Experimental 171
6.2. Marco de trabajo 178
6.3. Resultados 179

Capítulo 7. Conclusiones y líneas futuras **185**
7.1. Conclusiones 185
7.2. Líneas futuras 190

Bibliografía **193**

Apéndice A. "Mapping" de CIM a OWL **209**
A.1. Complejidad del modelo CIM 209
A.2. Expresión en $\mathcal{AL}\varepsilon\mathcal{CNOQ}^{-}_{\mathcal{H}R^{+}\circ}$ del Modelo Central de CIM 210
A.3. Especificación OWL del Modelo de Políticas CIM 220
A.4. Especificación $\mathcal{AL}\varepsilon\mathcal{CNOQ}^{-}_{\mathcal{H}R^{+}\circ}$ del Modelo de Políticas CIM 234

Apéndice B. Semántica Formal del lenguaje ACSL **243**
B.1. Descripción UML de la sintaxis abstracta 243
B.2. Sintaxis abstracta 244
B.3. Sintaxis XML (XML Schema) 247

Apéndice C. Especificaciones ACSL **255**
C.1. Protocolo [FIPA] IteratedContractNet 255
C.2. Protocolo para compensación de tareas 263
C.3. Protocolo de aprendizaje cooperativo de Sian 268

Apéndice D. Conceptualización DL de una holarquía **281**
D.1. Especificación en lógica descriptiva de un caso de aplicación 281
D.2. Especificación OWL del modelo de holarquía 295

Índice de figuras

2.1. Gestores y agentes OSI en una configuración de entorno en cascada . 23
2.2. Dominios de gestión y dominios administrativos en la gestión OSI . . 24
2.3. Modelos de gestión cooperativo y jerárquico definidos por la interfaz X 26
2.4. Elementos del meta-esquema CIM 30
2.5. Comparativa entre los modelos de información de las arquitecturas de gestión existentes y la propuesta (*Nesmarq*) 39
2.6. Comparativa entre los modelos de información de las arquitecturas de gestión existentes y la propuesta (*Nesmarq*) 40
2.7. Comparativa entre los modelos de organización de las arquitecturas de gestión existentes y la propuesta (*Nesmarq*) 41
2.8. Comparativa a nivel cualitativo entre de las arquitecturas de gestión existentes y la propuesta (*Nesmarq*) 42

3.1. Modelo de Información propuesto 49
3.2. Modelo de políticas CIM versión 2.6 (DMTF Service Level Agreements WG) 56
3.3. Jerarquía de herencia de las clases asociación presentes en el modelo de políticas CIM como instancias de la meta-clase ASSOCIATION (DMTF Service Level Agreements WG) 71
3.4. Modelo Central de CIM versión 2.6 (DMTF SysDev-WG) 80
3.5. Jerarquía de herencia de las clases asociación presentes en el *Modelo Central* de CIM como instancias de la meta-clase ASSOCIATION (DMTF SysDev-WG) 81

4.1. Especificación FSM del protocolo utilizado para negociar la asignación de tareas dentro de un holón de gestión [Fase de compensación] . 102
4.2. Especificación AUML del protocolo para negociar la asignación de tareas entre los miembros de un holón de gestión (Fase de asignación) 103
4.3. Especificación AUML del protocolo para negociar la asignación de tareas entre los miembros de un holón de gestión (Fase de compensación) 104

4.4. Representación textual de la especificación ACSL del protocolo Contract-Net . . . 107
4.5. Elementos de la sintaxis abstracta del lenguaje ACSL . . . 108
4.6. Elementos ACSL para la descripción de los intercambios de mensajes correlacionados . . . 109
4.7. Especificación AUML del protocolo de coordinación de Sian para dos agentes . . . 135
4.8. Especificación AUML del protocolo de coordinación de Sian para más de dos agentes . . . 136

5.1. Composición de una holarquía . . . 146
5.2. Composición de una holarquía orientada a roles . . . 148
5.3. Dominios de cooperación . . . 151
5.4. Interfaces de cooperación holónicas . . . 155
5.5. Arquitectura de un sistema para control de seguridad basado en SLA 156
5.6. Modelo de organización para un sistema para control de seguridad basado en SLA . . . 157
5.7. Modelo de organización para un sistema para control de seguridad basado en SLA (II) . . . 159

6.1. Entorno integrado de desarrollo $ACSL - AUML*$ . . . 180
6.2. Herramienta de modelado visual $CIMOnt$ . . . 181
6.3. Herramienta de modelado visual $MHolon$ . . . 182
6.4. Captura de una simulación de conversación multi-protocolo en la plataforma COMADA . . . 183

A.1. Modelo Central de CIM versión 2.6 (DMTF SysDev-WG) . . . 210
A.2. Jerarquía de herencia de las clases asociación presentes en el *Modelo Central* de CIM como instancias de la meta-clase ASSOCIATION (DMTF SysDev-WG) . . . 214
A.3. Modelo de políticas CIM versión 2.6 (DMTF Service Level Agreements WG) . . . 220
A.4. Jerarquía de herencia de las clases asociación presentes en el modelo de políticas CIM como instancias de la meta-clase ASSOCIATION (DMTF Service Level Agreements WG) . . . 221

B.1. Elementos de la sintaxis abstracta del lenguaje ACSL . . . 243
B.2. Elementos ACSL para la descripción de los intercambios de mensajes correlacionados . . . 244

C.1. Especificación AUML del protocolo de coordinación de Sian para más de dos agentes 271

D.1. Composición de una holarquía orientada a roles 283

Índice de cuadros

2.1. Especificaciones de los elementos fundamentales de la arquitectura de gestión de Internet SNMPv1 13
2.2. Especificaciones Fundamentales de la arquitectura de gestión de Internet SNMPv2 17
2.3. Especificaciones Fundamentales de la arquitectura de gestión de Internet SNMPv3 17
2.4. Principales propuestas del IETF para descentralizar el modelo organizativo 19
2.5. Modelo de información de gestión OSI 19
2.6. Relación entre las construcciones GDMO y los conceptos generales de orientación a objetos en que se apoyan 21
2.7. Comparación entre las MIBs de OSI e Internet 22
2.8. Estándares del OMG afines a la gestión de sistemas distribuidos . . . 31
2.9. Comparación de la expresividad de los modelos OMA 34

3.1. Reglas de traducción SMI-GDMO 54
3.2. Reglas de traducción GDMO-IDL 55
3.3. Reglas de traducción MIF-CIM 55
3.4. "Mapping" de esquemas CIM 61
3.5. "Mapping" de clases CIM 62
3.6. "Mapping" de jerarquías de generalización CIM 63
3.7. "Mapping" de propiedades CIM 64
3.8. "Mapping" de asociaciones CIM 67
3.9. Resumen del "mapping" CIM-OWL FULL 75
3.10. Descripción de los tipos de Lógicas Descriptivas utilizados y sus complejidades asociadas 78
3.11. Correspondencia entre la sintaxis OWL y $\mathcal{ALE}\mathcal{CNOQ}^{-}_{\mathcal{H}R^{+}\circ}$ 79
3.12. Síntesis del "mapping" CIM-$\mathcal{ALE}\mathcal{CNOQ}^{-}_{\mathcal{H}R^{+}\circ}$ 88

5.1. Expresiones de dominio en lógica descriptiva 152

6.1. Complejidad de la especificación CIM en términos de número de clases, atributos y relaciones . . . 172
6.2. Complejidad de la especificación CIM en términos de número de clases, atributos y relaciones (Cont.) . . . 173
6.3. Complejidad de la especificación CIM en términos de número de clases, atributos y relaciones (Cont.) . . . 174
6.4. Herramientas de razonamiento no evaluadas . . . 176
6.5. Herramientas de razonamiento evaluadas . . . 177
6.6. Protocolos de interacción evaluados . . . 177
6.7. Complejidad del modelo CIM . . . 180

A.1. Complejidad de la especificación CIM en términos de número de clases, atributos y relaciones . . . 211
A.2. Complejidad de la especificación CIM en términos de número de clases, atributos y relaciones (Cont.) . . . 212
A.3. Complejidad de la especificación CIM en términos de número de clases, atributos y relaciones (Cont.) . . . 213

D.1. Correspondencia entre la sintaxis LISP y la sintaxis abstracta DL . . 282

Lista de abreviaturas

ACL	Agent Communication Language
AN	Active Network
AUML	Agent UML (OMG)
ANSI	American National Standards Institute
AOP	Agent Oriented Programming
AOSE	Agent Oriented Software Engineering
ASN.1	Abstract Syntax Notation dot One
BER	Basic Encoding Rules
BNC	Backus Naur Form
ABNF	Augmented Backus Naur Form
CCITT	Comité Consultatif International Télégraphique et Téléphonique
CDR	Common Data Representation
CFP	Call For Proposals
CIM	Common Information Model (WBEM)
CMIP	Common Management Information Protocol (OSI)
CMIS	Common Management Information Services (OSI)
CMOT	CMIP Over TCP (OSI)
CORBA	Common Object Request Broker Architecture (OMA)
DAML	DARPA Agent Markup Language (DARPA-KSF)
DCE	Distributed Computing Environment
DCOM	Distributed Component Object Model
DEN	Directory Enabled Networks
DISMAN	Distributed Management Working Group (IETF)
DL	Description Logics
DMI	Definition of Management Information
DMI	Desktop Management Interface (DMTF)
DMTF	Distributed Management Task Force
DPE	Distributed Processing Environment
FIPA	Foundation for Intelligent Physical Agents
GDMO	Guidelines for the Definition of Managed Objects (OSI)
GIOP	Global Inter-ORB Protocol
GMOC	Generic Managed Object Class (OSI)
HEMS	High-Load Entity Management System (IETF)
HMM	Hyper-Media Management
HMMS	Hyper-Media Management Schema (HMM)
HMMP	Hyper-Media Management Protocol (HMM)
HMOM	Hyper-Media Object Manager (HMM)
HMP	Host Monitoring Protocol (IETF)
HTTP	Hyper-Text Transmission Protocol (IETF)

IDL	Interface Definition Language (OMA)
IETF	Internet Engineering Task Force
IIOP	Internet Inter-Orb Protocol (OMA)
IN	Intelligent Network
IOR	Interoperable Object Reference (CORBA)
IP	Internet Protocol (IETF)
ISO	International Standardization organization
ITU	International Telecommunications Union
JDMK	Java Dynamic Management Kit (JMAPI)
JIDM	Joint Inter-Domain Management
JMAPI	Java Management Application Programming Interface
KIF	Knowledge Interchange Formalism
KQML	Knowledge Query and Manipulation Language
KRSS	Knowledge Representation System Specification
KSE	Knowledge Sharing Effort (DARPA)
MAS	Multi-Agent System
MASIF	Mobile Agent System Interoperability Facility (OMG)
MIB	Management Information Base
MIF	Management Information Format (DMI)
MO	Managed Object
MOC	Managed Object Class (OSI)
MOF	Managed Object Format (WBEM)
MOF	Meta Object Facility (OMA)
ODL	Object Definition Language (TINA)
ODMA	Open Distributed Management Architecture
OMA	Object Management Architecture (OMG)
OMG	Object Management Group
ORB	Object Request Broker (OMA)
OSI	Open Systems Interconnection
OWL	Ontology Web Language (W3C)
RDF	Resource Description Framework (W3C)
RFC	Request For Comments (IETF)
RFP	Request For Proposals
RMI	Remote Method Invocation
RMON	Remote network MONitoring
RPC	Remote Procedure Call
SDL	Specification and Description Language
SGMP	Simple Gateway Monitoring Protocol (IETF)
SLA	Service Level Agreements
SMI	Structure of Management Information (IETF)
SNMP	Simple Network Management Protocol (IETF)
TCP	Transmission Control Protocol
TINA-C	Telecommunications Information Network Management Consortium

TME	Tivoli Management Environment
TMN	Telecommunications Management Network
UML	Unified Modeling Language (OMG)
URL	Universal Resource Locator
URI	Universal Resource Identifier
WBEM	Web Based Enterprise Management
XMI	Xml Metadata Interchange (OMA)
XML	eXtensible Markup Language (W3C)
XMP	X/Open Management Protocol
XSL-T	Xml Style-sheet Language Transformation (W3C)

Capítulo 1

Introducción

Índice General

1.1. Introducción y motivaciones 1
1.2. Contribución 7
1.3. Estructura de la tesis 8

1.1. Introducción y motivaciones

La creciente complejidad, heterogeneidad y dinamismo inherente a las redes de telecomunicaciones, los sistemas distribuidos y los servicios avanzados de información y comunicación actuales, así como el incremento de su criticidad e importancia estratégica, requieren la adopción de tecnologías cada vez más sofisticadas para su gestión, su control, su coordinación e integración por parte de los operadores de red, los proveedores de servicio y las empresas, como usuarios finales de los mismos, con el fin de garantizar niveles adecuados de funcionalidad, rendimiento y fiabilidad.

Las estrategias de gestión adoptadas tradicionalmente adolecen de seguir modelos excesivamente estáticos y centralizados, con un elevado componente de supervisión y difícilmente escalables. Así, los modelos clásicos de gestión estandarizados por ISO para OSI (OSI-MA) [Yem93] y la IETF para Internet (SNMP) [Ros96] sólo resultan adecuados en aplicaciones tradicionales de gestión de red, construidas bajo una perspectiva horizontal centrada tan solo en información relevante para el estado operativo de la red [Fei95]. Estas consideraciones se extienden a las plataformas existentes para gestión de telecomunicaciones como, por ejemplo, TMN de ITU [IT00], que han sido desarrolladas bajo esta perspectiva.

La acuciante necesidad por flexibilizar esta gestión y hacerla a la vez más escalable y robusta, ha provocado en los últimos años un elevado interés por utilizar nuevos

paradigmas basados en modelos jerárquicos y distribuidos, como evolución natural de los primeros modelos jerárquicos débilmente distribuidos [Wal97a] que sucedieron al paradigma centralizado. Se crean así nuevos modelos como son los basados en Gestión por Delegación [GY95, VPK97], en el paradigma de código móvil [FPV98] y en las tecnologías de objetos distribuidos. Fruto de esta línea de investigación son las Redes Inteligentes, las Redes Activas [TSS+97], la arquitectura *Open Distributed Management Architecture* (ODMA) de ISO [ISO95], la integración de CORBA en plataformas de gestión como TINA-C [PTBH98, Pav99] y las arquitecturas basadas en web como la *Web Based Enterprise Management* (WBEM) [DMT03] o la *Web-Based Integrated Management Architecture* (WIMA) [MF03]. Estas alternativas se han mostrado enormemente robustas, flexibles y escalables frente a las estrategias tradicionales de gestión, pero continúan sin resolver aún muchos problemas relacionados con la escalabilidad y el grado de autonomía de las soluciones de gestión desarrolladas en base a las mismas. Pese a ello, gran parte de las líneas actuales de investigación siguen decantándose por este último enfoque: utilización de tecnologías de objetos distribuidos y tecnologías Web conjuntamente con el paradigma jerárquico-distribuido [HAN98].

Las líneas actuales de investigación que parten del hecho de que muchos problemas de robustez, escalabilidad y flexibilidad continúan sin ser resueltos por el paradigma jerárquico-distribuido, abogan por la migración hacia un paradigma cooperativo fuertemente distribuido. Estas líneas [HB99, HB01] tienen su germen en la Inteligencia Artificial Distribuida (DAI) y, más concretamente, en el paradigma de agentes autónomos y en los Sistemas Multiagente (MAS). Todas ellas se perfilan en torno a un conjunto de objetivos que pueden resumirse en:

- Alcanzar un mayor grado de autonomía en la funcionalidad de la gestión y una mayor capacidad de autoconfiguración que resuelva los problemas de escalabilidad y la necesidad de supervisión presentes en los sistemas actuales,
- Evolucionar hacia técnicas de control fuertemente distribuido y cooperativo dirigido por la meta, y
- Dotar de una mayor riqueza semántica a los modelos de información.

Cada vez más investigadores están empezando a utilizar agentes para la gestión de redes y sistemas distribuidos. Sin embargo, los límites establecidos en sus trabajos entre agentes móviles (que siguen el paradigma de código móvil) y agentes autónomos (que realmente siguen el paradigma cooperativo) resultan difusos

[Mou98, ZCPZ97]. Muchos de estos trabajos se centran en la utilización de agentes móviles, lo cual, al igual que ocurría con las técnicas de código móvil comentadas anteriormente, les permite explotar el concepto de *Gestión por Delegación* [Yem93, VPK97][YEMI91] dinámicamente. Para ello, proponen la aplicación de nuevas técnicas que permiten dotar a los elementos de red de funcionalidad programable de forma dinámica (delegación dinámica de funcionalidad) [GY98, GPG00, PT00, BPC00, RD00, WPB99]. Con ello se consigue flexibilizar la gestión, distribuir la lógica de gestión cerca de los datos y distribuir el control. Sin embargo se permanece en el paradigma jerárquico distribuido. También se observan propuestas híbridas que promulgan la utilización de agentes orientados a objetos (distribuidos) en un intento, cuando menos discutible al tratarse de conceptualizaciones diferentes, de combinar las ventajas de ambos mundos [KJ98].

Si bien continúa sin definirse una arquitectura de gestión fiel al paradigma cooperativo fuertemente distribuido, estas líneas de investigación han puesto de manifiesto serios problemas de adecuación en los modelos de información, comunicación y organizativo (no así el funcional, que permanece inalterable desde la propuesta de ISO [Yem93]) de las arquitecturas de gestión existentes, por lo que urge proponer y estandarizar nuevas arquitecturas y marcos de trabajo que exploten el paradigma cooperativo en cada uno de ellos, así como desarrollar, en base a las mismas, plataformas en que sustentar las nuevas aplicaciones de gestión basadas en este nuevo paradigma. Hasta donde alcanza el conocimiento del autor, no existe ninguna propuesta completa de arquitectura fundamentada en el paradigma cooperativo (ni, por consiguiente, de marco de trabajo) que explote el paradigma de agentes autónomos en sus modelos (información, comunicación, organizativo y funcional). Esta opinión es compartida por otros autores (ver por ejemplo el análisis del espacio de soluciones que realiza [MF03], o la sección 4.20 de [Udu00]). Los principales obstáculos para la consolidación del paradigma cooperativo vienen determinados por la ausencia de técnicas y lenguajes de modelado y especificación adecuados a este paradigma que contemplen aspectos tales como la representación semántica de la información de gestión, la especificación de estados mentales (creencias, metas, responsabilidades, obligaciones, etc) asociados a los agentes de gestión, la especificación de comportamiento (protocolos de interacción, etc.), relaciones, estructuras organizativas, etc. en la construcción de los artefactos software del sistema.

En los últimos años se viene observando un creciente interés por la aplicación práctica del paradigma de agentes [IL02, WWC01, Wei99, HS97] en el modelado de sistemas distribuidos, principalmente en entornos heterogéneos y descentralizados como Internet. La autonomía y la sociabilidad inherente a los agentes se prometen

como elementos clave en el modelado de este tipo de sistemas. Puesto que el paradigma de agentes supone un cambio sustancial en los "artefactos" software de los sistemas a diseñar respecto de los existentes (principalmente respecto de los asociados al paradigma de objetos), se requiere la evolución de las herramientas de modelado existentes [IT95, BS97, RJB99, OMG02] para acomodar nuevos componentes que permitan visualizar, especificar, construir y documentar dichos "artefactos". Se requieren por tanto nuevos meta-modelos, nuevos modelos estructurales, organizativos, de comportamiento y de colaboración, nuevos lenguajes de especificación con semánticas apropiadas y nuevas notaciones tanto gráficas como sintácticas que faciliten la validación de dichos modelos.

A lo largo de los años se han desarrollado diferentes abstracciones y modelos con el fin de capturar la sociabilidad inherente a los agentes autónomos que participan en los sistemas distribuidos multi-agente, facilitando así el análisis y diseño de las, así denominadas, *sociedades de agentes*. Las propuestas fundamentadas en la extensión de otras técnicas tradicionales de análisis y diseño como, por ejemplo, las basadas en la orientación a objetos [Ken99], resultan poco adecuadas para la conceptualización de sociedades de agentes, debido fundamentalmente al fuerte distanciamiento existente entre las abstracciones implicadas. Los métodos composicionales utilizados tradicionalmente en arquitecturas software orientadas a objetos [BDKJT97] ofrecen escasa aplicabilidad en la definición de las estructuras comunicativas, organizativas y orientadas a roles que caracterizan las sociedades de agentes.

En los últimos años se han propuesto un conjunto de técnicas de modelado específicas para el paradigma de agentes, algunas de las cuales empiezan a concebir los sistemas multi-agente como organizaciones artificiales o instituciones. En la mayoría de los casos, estas técnicas de modelado definen una organización como una colección de roles (y por tanto un modelo de roles) [FERB98] y un conjunto de protocolos de interacción que ocurren entre dichos roles [DC96], sin aportar abstracciones organizativas de más alto nivel (puestos de trabajo, dominios, organizaciones, instituciones, normas, responsabilidades, obligaciones, leyes sociales o reglas orgánicas) que ayuden en la compleja tarea de modelar una sociedad de agentes.

Recientemente, se observa cada vez más una tendencia generalizada por desarrollar abstracciones organizativas que permitan ver los sistemas multi-agente como organizaciones abiertas (y posiblemente distribuidas). La metáfora social y los modelos basados en roles empiezan a ser generalmente aceptados como una manera natural de concebir los sistemas multi-agente. Además, las propuestas iniciales han madurado tremendamente hasta asentarse en [Omi01] y [ZAMB01]. La concepción que este último hace de las organizaciones como marcos de trabajo donde ubicar

una topología basada en el concepto de rol [BT79], ha servido de revulsivo para que se adopte esta metáfora en otras propuestas posteriores como [OPF03] o [SRA02, DdS02]. Esto se debe a que tal modelo permite restringir el comportamiento de los agentes en una organización (institución) a través de una serie de reglas/normas, deducir a partir de ellas las estructuras existentes en la organización e incluso extraer o explotar ciertos patrones de comportamiento.

En paralelo con estos modelos y abstracciones han surgido un conjunto de metodologías de desarrollo específicas del paradigma de agentes y que hacen uso de los mismos. Cabe citar por su actualidad e interés las recogidas en [WWC01, IGG99] y [CIAN01]. Otras metodologías como CoMoMAS [GLAS02] están totalmente orientadas a la ingeniería del conocimiento, y no a la ingeniería del software. Los modelos y metodologías propuestos comparten un amplio conjunto de conceptos comúnmente aceptados, pero carecen de un lenguaje y una semántica con que expresar formalmente dichos conceptos. Esta laguna impide a muchos usuarios la entrada en el mercado de la tecnología de agentes (es el caso de los usuarios del dominio de gestión distribuida) y el modelado de agentes, así como la expansión del soporte al modelado aportado por herramientas CASE, hasta ahora prácticamente inexistentes.

Hasta la fecha, y hasta donde alcanza el conocimiento del autor, no existe ningún lenguaje que permita especificar, construir, visualizar y documentar los artefactos de un sistema software basado en agentes y que resulte adecuado para el dominio de la gestión cooperativa de sistemas y servicios distribuidos. La única propuesta significativa, Agent-UML [BMO, JOB00, OPB00b], tan solo supone un refinamiento de la notación UML [RJB99, OMG01a], principalmente de sus diagramas de interacción tipo secuencia, pero no aporta ninguna formalización semántica propia (pese a que la semántica UML resulta en muchos casos poco apropiada en este contexto) y adolece de contemplar muchos aspectos del nuevo paradigma, si bien se está trabajando en este sentido y así lo demuestran trabajos en la línea de [PO01]. Otras propuestas de marcos de especificación de agentes como Agents in Z [LUCK01] o Desire [BDKJT97] se alejan del planteamiento de la propuesta de esta tesis al ser esencialmente adaptaciones de técnicas de ingeniería de conocimiento. Sin embargo, nos encontramos en un momento clave para la elaboración de herramientas de modelado adecuadas para el paradigma de agentes, propiciado por la convergencia de numerosos modelos de abstracción y metodologías como las comentadas anteriormente, así como de arquitecturas abstractas (e.g. [FIP02]) que están alcanzando un significativo grado de madurez, y de las cuales pueden extraerse los principales artefactos presentes en los sistemas distribuidos basados en agentes sobre los que existe ya un elevado consenso.

En paralelo con el trabajo llevado a cabo en el área de la organización de los sistemas multi-agente, se viene desarrollado un modelo organizativo denominado "holónico" que ha sido principalmente utilizado en planificación y control distribuido de sistemas de manufactura y producción [MCFA00] y en el modelado de empresas virtuales [UWB01]. El concepto de holón (término que se compone de la palabra griega "holos", que significa "el todo" y el sufijo "on" que evoca una partícula o "parte") está inspirado en la observación de las estructuras recursivas autosimilares presentes en los sistemas organizativos biológicos y sociales. Hace 25 años, su inspirador, el filosofo húngaro Arthur Koestler [Koe68], definió el término holón como una parte identificable de un sistema que tiene identidad única, compuesta de partes subordinadas y que, a su vez, es parte de un todo. El término holón se emplea para denominar entidades que exhiben simultáneamente un comportamiento autónomo (se comportan como un todo) y no autosuficiente (se comportan como una parte de un todo mayor), por lo que requieren de mecanismos efectivos de comunicación y coordinación. Los estrictos requisitos de predecibilidad y tiempo real asociados al entorno (informático) en que viene utilizándose predominantemente el modelo holónico ha provocado una notable resistencia a relacionar dicho modelo con el paradigma de agentes autónomos [FLET00], pese a que este modelo resulta especialmente adecuado para representar estructuras organizativas heterárquicas (cooperativas). No obstante, se observa un creciente interés por esta nueva línea de investigación y existe ya un workshop internacional de carácter específico orientado a la utilización conjunta de los sistemas multiagente y el modelo de organización holónico [HOLO02]. Sin embargo, el ámbito de aplicación del modelo de organización holónico apenas si ha trascendido al área de gestión de sistemas y servicios distribuidos.

Por otra parte, el modelado de la información de gestión, y su estrecha relación con las capacidades de automatización de las soluciones de gestión elaboradas, demanda cada vez con más fuerza la utilización de nuevos formalismos que faciliten la captura de la semántica de los modelos, así como su transmisión, su comunicación y el razonamiento automático acerca de los mismos. La *representación del conocimiento* y el *modelado conceptual* [DJM$^+$00] representan las áreas de la inteligencia artificial que mayores avances ha conseguido en este respecto, sin embargo, apenas si ha tenido incidencia en ninguno de los modelos de información de gestión elaborados hasta la fecha. La representación del conocimiento, como área de investigación, se centra en el diseño de formalismos adecuados a nivel epistemológico y a nivel computacional para expresar el conocimiento adquirido acerca de un dominio particular (en el caso que nos ocupa, el dominio de gestión). Una de las principales líneas de in-

vestigación en este respecto es la relacionada con el principio de que el conocimiento puede representarse mediante la caracterización de los conceptos presentes en el dominio y las relaciones existentes entre dichos conceptos. A partir de estas premisas, se han elaborado numerosas propuestas de representación como son las *redes semánticas* [Qui66], los marcos de Minsky [Min75, FK85] y, más recientemente las lógicas descriptivas [McG03] y los lenguajes de ontologías [CvHH+01, Hor02, WG03b], para las que comienzan a proponerse metodologías [FGPPP99]. Tomando como base los avances conseguidos en este terreno por la comunidad internacional de inteligencia artificial, cabe reconsiderar la línea seguida en la elaboración de los modelos de información de gestión tradicionales y estudiar la posibilidad de incorporar técnicas relacionadas con el área de representación del conocimiento, así como los beneficios derivados de dicha decisión.

En este último sentido, el *Common Information Model* (CIM) constituye el formalismo estándar para el modelado de información de gestión desarrollado por el *Distributed Management Task Force* (DMTF), en el contexto de su propuesta WBEM [DMT03], para representar una vista conceptual del entorno gestionado. Existe un consenso generalizado en la necesidad de dotar a los diagramas CIM de una semántica precisa que permita establecer un entendimiento común del significado formal de las construcciones del meta-modelo CIM utilizadas, con el propósito de facilitar la interoperación y la cooperación. Esta necesidad ha sido reconocida recientemente por el propio DMTF en un "keynote" presentado en la conferencia internacional *IEEE Policy 2003* [Wes03]. Sin embargo, hasta la fecha, y hasta donde alcanza el conocimiento del autor, no se ha realizado ninguna propuesta concreta de formalización de los modelos CIM. Si bien existen propuestas de formalización de los diagramas estructurales UML [Eva97, Eva98] fácilmente adaptables a los diagramas CIM, ninguna de estas propuestas supone un fundamento sólido para el desarrollo de técnicas de razonamiento automático, basadas en algoritmos correctos y completos con respecto a la semántica.

1.2. Contribución

En este contexto, este trabajo de Tesis presenta un modelo de arquitectura para gestión holónica de sistemas y servicios distribuidos mediante sociedades de agentes autónomos, cuyos objetivos fundamentales pueden resumirse en:

- El incremento del grado de automatización asociado a las tareas de gestión,
- El aumento de la escalabilidad de las soluciones de gestión elaboradas en base a la misma,

- La provisión de soporte para delegación tanto por dominios como por macrotareas, y
- Un alto grado interoperabilidad en entornos abiertos.

A partir de estos objetivos se ha desarrollado un *modelo de información* formal de tipo semántico, basado en *lógica descriptiva* que permite un mayor grado de automatización en la gestión en base a la utilización de agentes autónomos racionales, capaces de razonar, inferir e integrar de forma dinámica conocimiento y servicios conceptualizados mediante el modelo CIM y formalizados a nivel semántico mediante lógica descriptiva. El modelo de información propuesto permitirá además desarrollar herramientas CASE de última generación con soporte formal para la validación (en términos de coherencia, satisficibilidad, etc.) de los modelos conceptuales CIM elaborados mediante las mismas.

El modelo de información incluye además un "mapping" a nivel de meta-modelo de CIM al lenguaje de especificación de ontologías OWL, que supone un significativo avance en el área de la representación y el intercambio basado en XML de modelos y meta-información.

A nivel de interacción, el modelo aporta un lenguaje de especificación formal de conversaciones entre agentes basado en la teoría de actos ilocucionales y aporta una semántica operacional para dicho lenguaje que facilita la labor de verificación de propiedades formales asociadas al protocolo de interacción como pueden ser su terminación en tiempo finito, la alcanzabilidad de sus estados, etc.

Se ha desarrollado también un *modelo de organización* holónico y orientado a roles cuyas principales características están alineadas con las demandadas por los servicios distribuidos emergentes e incluyen la ausencia de control central, capacidades de reestructuración dinámica, capacidades de cooperación, y facilidades de adaptación a diferentes culturas organizativas. El modelo incluye un submodelo normativo adecuado al carácter autónomo de los holones de gestión y basado en las lógicas modales deontológica y de acción.

1.3. Estructura de la tesis

En el capítulo 2 se proporciona una visión de conjunto de las diferentes arquitecturas de gestión existentes desde la perspectiva de la adecuación de sus modelos de información y de organización a las nuevas necesidades demandadas por los sistemas y servicios distribuidos emergentes. El capítulo concluye con un análisis comparativo del espacio de soluciones realizado por el autor como aportación personal al estado de la cuestión en arquitecturas de gestión.

Los capítulos 3 y 4 presentan el modelo de información propuesto. El capítulo 3 se centra en la parte estructural de dicho modelo y describe un "mapping" altamente expresivo a nivel de meta-esquema del lenguaje de modelado CIM desarrollado por el DMTF al lenguaje de ontologías OWL para, a continuación, adaptar este "mapping" a una lógica descriptiva computable, decidible y completa. El capítulo 4 se centra en la parte dinámica o de comportamiento del modelo de información y presenta un lenguaje de especificación formal de conversaciones, basado en la teoría de actos hablados, así como una semántica operacional para dicho lenguaje basada en un sistema transicional. Esta semántica proporciona una descripción sin ambigüedades de la ejecución de las diferentes construcciones del lenguaje y puede facilitar trabajos adicionales relacionados con la validación de propiedades formales asociadas a la especificación de un protocolo de interacción.

El capítulo 5 está dedicado a la propuesta de un modelo de organización basado en los conceptos de holarquía, rol, dominio de cooperación, así como a la propuesta de un modelo normativo adecuado a dicho modelo.

El capítulo 6 presenta un diseño experimental adecuado para la validación de los modelos propuestos en los tres capítulos anteriores y describe el marco de trabajo que ha sido desarrollado para facilitar la experimentación práctica y la obtención de resultados acerca de dichos modelos según el diseño experimental descrito.

El capítulo 7 ofrece las conclusiones y resultados derivados del trabajo realizado, y se sugieren posibles líneas futuras de investigación relacionadas con la propuesta.

Los apéndices que se incluyen recogen la sintaxis abstracta y la sintaxis XML del lenguaje ACSL presentado en el capítulo 4, así como diversos casos de aplicación de cada una de las propuestas realizadas en la tesis y que sirven como validación de las mismas.

Capítulo 2

Estado de la cuestión

Índice General

2.1. La arquitectura de gestión de Internet 12
2.2. La arquitectura de gestión de OSI 18
2.3. La arquitectura DMI del DMTF 26
2.4. La arquitectura WBEM del DMTF 28
2.5. OMA como arquitectura de gestión 31
2.6. Gestión basada en políticas PBN 36
2.7. Análisis de las arquitecturas existentes 38

El profundo proceso de reestructuración que está aconteciendo en el mercado de las telecomunicaciones y las redes de datos introduce nuevas necesidades de interoperación y nuevas relaciones estratégicas entre las diferentes partes implicadas en la provisión de los nuevos servicios telemáticos emergentes. La creciente complejidad y sofisticación de estos servicios requiere a su vez de un grado cada vez mayor de automatización en su gestión. Este hecho ha motivado el estudio comparativo de las diferentes arquitecturas de gestión existentes desde la perspectiva de

- La riqueza semántica de la información utilizada en su comunicación por los diferentes elementos participantes. Para que los diferentes elementos de un sistema de gestión puedan comunicarse entre sí deben compartir un lenguaje y un vocabulario común (modelo de información) con los que expresar su conocimiento de gestión, así como un protocolo de gestión con que transportar dicha información. Estas dos partes constituyen la interfaz estándar a través de la cual ocurre la comunicación.
- Sus capacidades para relacionarse, cooperar y negociar de una forma organizada y coherente para la provisión y gestión de dichos servicios. La funcionalidad

de un sistema de gestión está limitada tanto por el poder expresivo de su modelo de información y las posibilidades del protocolo de gestión utilizado en la comunicación, como por el modelo organizativo seguido. Este último determina los roles desempeñados por los diferentes elementos del sistema, cómo se pueden agrupar estos roles y las relaciones y formas de cooperación permitidas entre los mismos.

Las siguientes secciones presentan los modelos y arquitecturas de gestión existentes haciendo especial hincapié en cómo contemplan estos aspectos. El propósito es identificar las carencias existentes de modo que éstas puedan ser contempladas en el desarrollo de una arquitectura de gestión adecuada a las nuevas necesidades de interoperación, escalabilidad, automatización y cooperación.

2.1. La arquitectura de gestión de Internet

La arquitectura de gestión de Internet tiene sus orígenes en los primeros protocolos de gestión aparecidos a finales de los años 80: Host Monitoring Protocol (HMP), High-Load Entity Management System (HEMS), Simple Gateway Monitoring Protocol (SGMP) y, como evolución de este último, Simple Network Management Protocol (SNMP)[CFSD90]. Finalmente, el IETF adoptó este último como protocolo de gestión de la arquitectura que lleva su nombre. El modelo de gestión de Internet se utiliza predominantemente para gestionar redes de datos basadas en la arquitectura de comunicaciones TCP/IP.

Los intentos por basar la gestión de Internet en la arquitectura de gestión de OSI, utilizada por entonces en el entorno de las redes de telecomunicaciones, dieron como fruto la propuesta CMOT (CMIP Over TCP, [RFC1006]), consistente en la adaptación de su protocolo de gestión (CMIP) [IT91a] y sus servicios (CMIS) [IT91b] a la arquitectura de comunicaciones TCP/IP. Sin embargo, esta propuesta apenas ha tenido acogida fuera del contexto de las redes de telecomunicaciones.

El cuadro 2.1 recoge los principales documentos RFC relacionados con la especificación de la arquitectura de gestión de Internet (SNMPv1).

2.1.1. Riqueza semántica de la información de gestión

Dentro de la arquitectura de gestión de Internet, la información de gestión ofrece una perspectiva general del tipo y la estructura de la información intercambiada entre los componentes de un sistema de gestión para llevar a cabo las tareas de gestión. El modelo de información de Internet define la estructura de la base de

Documento	**Título**
RFC 1155	Structure and Identification of Management Information for TCP/IP Based networks
RFC 1156	Management Information Base for Network Management of TCP/IP Based Internets
RFC 1157	A Simple Network Management Protocol
RFC 1212	Concise MIB Definitions
RFC 1213	Management Information Base for Network Management of TCP/IP Based Internets: MIB-II

Cuadro 2.1: Especificaciones de los elementos fundamentales de la arquitectura de gestión de Internet SNMPv1

información de gestión de Internet, sus reglas de nombrado, su esquema organizativo y la notación utilizada para la definición de los objetos gestionados.

Los *objetos gestionados* [1] abstraen las características de los recursos gestionados relevantes para el proceso de gestión. En el modelo de información de Internet, estos objetos se modelan como simples variables con algunas características significativas adicionales tales como su tipo de datos, su identificador de objeto, su descripción y atributos de sólo lectura o lectura-escritura.

La *base de información de gestión* (MIB) de Internet [MR90] es una colección estructurada, de carácter lógico, de definiciones de objetos gestionados. Con el fin de conseguir interoperabilidad en la gestión se precisa un esquema común de representación, identificación y organización de los objetos de la MIB. Con este propósito, se especifica la *estructura de información de gestión* (SMI) de Internet [RM90], que identifica y restringe los tipos de datos que pueden utilizarse en la MIB, describe una técnica estandarizada para definir los objetos gestionados y la propia estructura de la MIB, determina una convención de nombrado para identificar los objetos gestionados de la MIB y especifica la técnica de codificación de los valores de dichos objetos. El uso de una SMI y un esquema de nombrado estandarizados extiende las posibilidades de interoperabilidad a las MIBs propietarias (extensiones privadas de la MIB de Internet).

La filosofía de diseño de la SMI de Internet fomenta la simplicidad, extensibilidad e interoperabilidad en la MIB en detrimento de la flexibilidad y funcionalidad. Es por ello que la MIB sólo puede almacenar como objetos gestionados tipos de datos escalares, vectores de escalares y matrices bidimensionales de escalares. No se admite

[1] El término *objeto gestionado* utilizado en el contexto de gestión SNMP difiere conceptualmente del término *objeto gestionado* utilizado en gestión de OSI y, en general, del concepto de *objeto* utilizado en la orientación a objetos.

anidamiento, esto es, que un elemento de una tabla se defina a su vez como un elemento tabular. Del mismo modo, SMI sólo permite la creación y el envío de escalares, incluidas entradas individuales en tablas, en contraste con la filosofía seguida en la gestión OSI que, como se verá más adelante, permite la creación y el envío de estructuras de datos complejas.

Cada objeto gestionado de la MIB se define formalmente en notación ASN.1 como una instancia de una definición de macro que especifica su tipo de datos (sintaxis), su forma de acceso y rango de valores permitidos (incluido el valor por omisión), así como su identificador de objeto y su relación (jerárquica) con otros objetos de la MIB.

La MIB de Internet presenta una estructura jerárquica en forma de árbol definida en ASN.1. Las hojas del árbol son los objetos gestionados, cada uno de los cuales representa [parte de] un recurso, actividad o información relacionada que debe ser gestionada. La estructura del árbol define por si misma una agrupación de objetos en conjuntos lógicamente relacionados.

Cada tipo de objeto en la MIB tiene asociado un identificador de objeto único que sirve para nombrarlo. La convención de nombrado es jerárquica, por lo que este identificador sirve también para identificar la estructura de tipos de objetos.

A nivel operacional, un gestor de red puede monitorizar el estado de los recursos de un sistema leyendo los valores de los objetos de la MIB y puede controlarlos únicamente modificando dichos valores. A esta aproximación al control se la suele denominar *MIB de instrumentación* ya que el control se produce como efecto colateral a la modificación del valor de los objetos de la MIB mediante la PDU de protocolo *SNMP_SetRequest*. El bajo nivel de abstracción proporcionado por esta aproximación, y su fuerte dependencia del protocolo de gestión subyacente, denota un escaso poder expresivo del modelo de información frente al aportado por esquemas basados en *lenguajes de definición de interfaces* (IDLs) y en mecanismos de *llamada a procedimiento remoto* (RPC). Por otra parte, el modelo de MIB SNMP es limitado y no facilita el desarrollo de aplicaciones que requieran consultas sofisticadas.

Los traps SNMP, al contrario que las notificaciones OSI suponen un conjunto reducido de mensajes de alarma asíncronos que no puede ser definido como específicos de un objeto de gestión.

SNMPv2 introduce una serie de mejoras tanto en la SMI como en las operaciones de protocolo. La SMIv2 permite elaborar más la especificación y la documentación de los objetos gestionados y la MIB. Para ello introduce cuatro conceptos clave que mejoran las capacidades de modelado de SMIv1: definiciones de objetos mejoradas con nuevos tipos de datos y mayor riqueza semántica, tablas conceptuales con faci-

lidades para creación/eliminación de filas, definiciones de notificaciones y módulos de información. El cambio más importante en las operaciones de protocolo lo constituye la inclusión de dos nuevas PDU: la PDU *SNMP_ GetBulkRequest* aumenta la eficiencia en la transmisión de grandes volúmenes de información de gestión, principalmente en la obtención de múltiples filas de una misma tabla, mientras que la PDU *SNMP_ InformRequest* permite el envío de alarmas entre gestores.

SMIv2 consigue mejorar la expresividad semántica de SMIv1 mediante la incorporación de convenciones textuales, que permiten una especificación más formal del comportamiento y la reutilización de las especificaciones.

2.1.2. Capacidades de organización

El modelo organizativo de la arquitectura de gestión de internet SNMPv1 es de tipo centralizado y está basado en el modelo de cooperación *gestor-agente* el cual, a su vez, se fundamenta en la arquitectura de comunicaciones *cliente-servidor*. El modelo se caracteriza por la existencia de un único gestor, que concentra la totalidad de la aplicación de gestión, y un conjunto de agentes, cuya funcionalidad prácticamente se limita a la mera recolección de información de acuerdo con la MIB asociada. El modelo asume implícitamente que todos los agentes deben ser gestionados de forma semejante, independientemente de la potencia de cálculo del equipo en que ejecuten. Además, si bien no existe ningún aspecto inherente a la especificación de SNMP que limite la utilización de alarmas, la gestión basada en SNMP utiliza predominantemente el modelo de sondeo para la gestión regular frente al modelo de notificación asíncrona, que relega a la gestión "ad-hoc". Esta última se realiza de modo reactivo y en gran medida sin automatización. Sus principales debilidades son por tanto su fragilidad y su falta de escalabilidad.

Con la aparición de las *MIBs para monitorización remota* RMON [Wal95] en 1991 y RMONv2 [Wal97b] en 1997, el IETF apuesta firmemente por flexibilizar el modelo organizativo de la gestión de Internet. RMON supone la primera apuesta por distribuir la gestión de redes TCP/IP al incorporar técnicas de gestión por delegación, con el propósito de automatizar en cierta medida la labor de gestión y reducir el efecto en el ancho de banda ocasionado por el tráfico de gestión. Como consecuencia, en RMON, un agente realiza y controla de forma autónoma labores de recolección, preprocesamiento y evaluación de información de gestión sin intervención/supervisión de un gestor. Sin embargo, la autonomía es mínima y se reduce a aspectos de monitorización ya que el agente es incapaz de tomar la iniciativa (proactividad) si se pierde la conexión con el gestor. El método de delegación es poco flexible y excesivamente estático, si bien puede considerarse que, en cierta me-

dida, un agente RMON realiza las tareas propias de algunas funciones de gestión de sistemas (SMFs) en OSI.

Si bien con la aparición de la MIB RMON en 1991 se observa un primer intento de cambio en el modelo organizativo de la gestión de Internet, al permitirse la distribución de parte de la funcionalidad de gestión a los agentes, el modelo continúa siendo excesivamente rígido ya que los agentes RMON son gestionados por una estructura plana de gestores. No es hasta la aparición de SNMPv2 en 1993 que se inicia la evolución hacia un modelo distribuido más flexible de tipo jerárquico. Con la propuesta de una nueva MIB para la comunicación entre gestores SNMPv2 (*Manager-to-Manager MIB*) [CMRW93] en abril de 1993 y la inclusión de una nueva PDU *SNMP_InformRequest* destinada a que un gestor intermedio pueda enviar alarmas asíncronas a un gestor jerárquicamente superior, SNMPv2 permite adoptar estrategias jerárquicas débilmente distribuidas. En este último caso, algunos sistemas operan como gestores intermedios con roles de agente y gestor simultáneamente. En su rol de agente, estos sistemas aceptan comandos provenientes de un sistema de gestión jerárquicamente superior; estos comandos podrían suponer el acceso a información almacenada localmente en el gestor intermedio, o requerir a este último la recolección y el resumen de información acerca de sus agentes subordinados (que de nuevo podrían ser a su vez gestores intermedios). Sin embargo, la fragilidad del modelo de seguridad adoptado por SNMPv2 dificultaron la implantación de estas nuevas capacidades organizativas destinadas a la gestión de la infraestructura de comunicaciones TCP/IP de empresas geográficamente dispersas.

Tras varios años de experiencia con SNMPv2, el IETF decidió revisar la especificación. Como consecuencia, en 1996 vieron la luz un grupo de RFCs que eliminaban los aspectos de seguridad introducidos con SNMPv2, cuyo planteamiento era a todas luces erroneo, e introducían algunas otras modificaciones menores. A este marco administrativo para SNMPv2 se le denominó SNMPv2c (*Community-based SNMPv2*) [SCM+96a]. Esta decisión facilitó la adopción de SNMPv2 consolidando las numerosas mejoras funcionales introducidas en su especificación inicial. Sin embargo, SNMPv2c soportaba tan solo un modelo centralizado excesivamente orientado a monitorización al adolecer de un modelo de seguridad sólido.

Los aspectos de seguridad se retoman en el mismo 1996 y, de entre todas las propuestas originadas, dos reciben especial atención: SNMPv2u y SNMPv2*. Estas dos propuestas sirven como punto de partida en 1997 para un nuevo grupo de trabajo *SNMPv3 Working Group* aprobado por el IETF en marzo de ese mismo año. En enero de 1998 este nuevo grupo produce un conjunto de propuestas de estandarización (ver cuadro 2.3) que definen un marco de trabajo para incorporar

Documento	**Título**
RFC 1901	Introduction to Community-Based SNMPv2
RFC 1902	Structure of Management Information for SNMPv2
RFC 1903	Textual Conventions for SNMPv2
RFC 1904	Conformance Statements for SNMPv2
RFC 1905	Protocol Operations for SNMPv2
RFC 1906	Transport Mappings for SNMPv2
RFC 1907	Management Information Base for SNMPv2
RFC 1908	Coexistence Between Version 1 and Version 2 of the Internet-Standard Network Management Framework

Cuadro 2.2: Especificaciones Fundamentales de la arquitectura de gestión de Internet SNMPv2

Documento	**Título**
RFC 2271	An Architecture for Describing SNMP Management Frameworks
RFC 2272	Message Processing and Dispatching for SNMP
RFC 2273	SNMPv3 Applications
RFC 2274	User-Based Security Model for SNMPv3
RFC 2275	View-Based Acces Control Model (VACM) for SNMP
RFC Draft	Introduction to Version 3 of the Internet Network Management Framework

Cuadro 2.3: Especificaciones Fundamentales de la arquitectura de gestión de Internet SNMPv3

seguridad tanto a SNMPv1 como a SNMPv2 [BW98, WPM98].

SNMPv3 por si solo es, como se expresa en el documento "Introduction to Version 3 of the Internet Network Management Framework", simplemente SNMPv2 más seguridad y administración. Sin embargo, abre de nuevo las puertas a la evolución hacia un modelo organizativo jerárquico fuertemente distribuido [LMS98].

Si bien SNMPv2 y las MIBs *Manager-to-Manager* y RMON supusieron conjuntamente un primer intento por evolucionar hacia modelos organizativos jerárquicos y distribuidos, no es hasta después de la formulación de la propuesta final de un modelo de gestión por delegación realizada por Goldszmidt en 1995 [GY95], que el IETF se plantea evolucionar más seriamente hacia un modelo organizativo jerárquico fuertemente distribuido que incorporase el concepto de gestión por delegación en toda su magnitud. Con este propósito se crea el grupo de trabajo *DISMAN IETF Working group*, entre cuyos resultados más significativos se encuentra la especificación de la *Script MIB* en 1999 [LS99], como parte de una propuesta global de marco

de trabajo para gestión distribuida (*Distributed Management Framework*, DMF).

La propuesta DMF supone la apuesta más clara del IETF por incorporar técnicas dinámicas de gestión por delegación en el modelo de gestión de Internet, apropiadas para sistemas distribuidos dinámicos y entornos organizativos dinámicos. La idea subyacente en el modelo organizativo propuesto es que ahora el gestor distribuye trabajos en forma de "scripts" bajo credenciales a un gestor distribuido que las ejecuta de forma distribuida en entidades denominadas *objetivos de gestión*. La propuesta DISMAN-DMF incluye la recomendación de un conjunto de servicios para formación de dominios, descripción de los sistemas destinatarios de la delegación, etc, así como la especificación de un conjunto de MIBs. Entre las MIBs propuestas destacan la *SCRIPT-MIB*, para la gestión del ciclo de vida de los "scripts", la *DISMAN-SERVICES-MIB* para la descripción de los servicios, la *TARGET-MIB* para la descripción de los destinos, la *SCHEDULE-MIB* para la planificación de las operaciones y la MIB de sucesos *EVENT-MIB*, basada en la MIB RMON y en la MIB gestor-gestor de SNMPv2 y que constituye un sucesor de ambas.

Si bien la propuesta DISMAN-DMF no está ligada explícitamente a una versión concreta del protocolo de gestión SNMP, se la suele asociar a SNMPv3 debido a las necesidades de seguridad implicadas en su utilización.

El cuadro 2.4 recoge las principales propuestas del IETF para descentralizar el modelo organizativo de la arquitectura de gestión de Internet en busca de una mayor robustez, escalabilidad y un mejor aprovechamiento del ancho de banda.

2.2. La arquitectura de gestión de OSI

La importancia adquirida por la arquitectura de gestión de OSI (ISO 7498-4) [IT92a] se debe a dos razones fundamentales. Por una parte, al igual que ocurre con el resto de la arquitectura de comunicaciones desarrollada por ISO, sirve como modelo de referencia a la hora de describir el resto de arquitecturas. Por otra parte, la arquitectura como tal ha tenido una gran aceptación en el área de las redes de telecomunicaciones, como muestra el hecho de que la arquitectura TMN (*Telecommunications Management Network*) [IT92d] para gestión distribuida de infraestructuras de telecomunicaciones esté basada en ella.

2.2.1. Riqueza semántica de la información de gestión

Al igual que ocurría con el modelo de información de Internet, el modelo de información de OSI se basa en los conceptos de *estructura de información de gestión*, *objeto gestionado* y *base de información de gestión*. El modelo define la estructura

Documento	**Título**
RFC 1757	Remote Network Monitoring Management Information Base (RMON)
RFC 2021	Remote Network Monitoring Management Information Base (RMONv2)
RFC 1451	Manager-to-Manager MIB
RFC 2925	Definitions of Managed Objects for Remote Ping, Traceroute, and Lookup Operations (DISMAN-DMF)
RFC 2982	Distributed Management Expression MIB (DISMAN-DMF)
RFC 2981	Event MIB (DISMAN-DMF)
RFC 3014	Notification Log MIB (DISMAN-DMF)
RFC 3165	Definitions of Managed Objects for the Delegation of Management Scripts (DISMAN-DMF)
RFC 3231	Definitions of Managed Objects for Scheduling Management Operations(DISMAN-DMF)

Cuadro 2.4: Principales propuestas del IETF para descentralizar el modelo organizativo

Estructura de información de gestión	**Especificación**
Modelo de información de gestión	X.720
Definición de la información de gestión	X.721
Directrices para la definición de objetos gestionados (GDMO)	X.722

Cuadro 2.5: Modelo de información de gestión OSI

de la base de información de gestión de OSI, sus reglas de nombrado, su esquema organizativo y la notación utilizada para la definición de sus objetos gestionados. El cuadro 2.5 muestra las diferentes partes en que divide ISO la especificación de su estructura de información de gestión.

Un *objeto gestionado* representa una abstracción software de un recurso implicado en la tarea de gestión. Viene caracterizado por un conjunto de atributos que representan las propiedades del recurso relevantes para el proceso de gestión, y un conjunto de operaciones que representan su comportamiento de gestión. El sistema de gestión percibe los recursos de red e interactúa con ellos únicamente a través de los objetos gestionados que los representan.

El concepto de objeto gestionado de OSI se fundamenta en el modelado de información orientado a objetos y aplica los conceptos genéricos de objeto, clase, encapsulación, herencia, asociación, agregación y polimorfismo al modelado de información de gestión de red. La orientación a objetos de OSI supone una aproxima-

ción arriba-abajo al diseño y promueve por tanto la reutilización de especificaciones e implementación.

Una *base de información de gestión* (MIB) OSI es una agrupación lógica de objetos gestionados que define su estructura (sintáctica y semántica), su nombrado y su organización. La estructura de una MIB OSI viene determinada por cuatro estructuras jerárquicas:

- Una jerarquía de registro de las clases de objetos gestionados y sus componentes que permite que estos puedan ser referenciados y utilizados por las aplicaciones de gestión.
- Una jerarquía de herencia que representa las relaciones de generalización existentes entre las clases de objetos gestionados.
- Una jerarquía de contención entre clases de objetos gestionados que facilita el modelado de objetos por composición.
- Un árbol de nombrado, instancia de la jerarquía de contención, que representa las relaciones de contención existentes entre instancias de objetos.

A diferencia de la MIB de Internet, la estructura de la MIB de OSI, también denominada *árbol de información de gestión* (MIT), viene determinada por el árbol de nombrado y no por la jerarquía de registro (única herramienta de modelado existente en la arquitectura de gestión de Internet). Adicionalmente, el árbol de registro de la MIB de Internet no distingue entre clases de objetos e instancias de objetos.

OSI define la estructura de las clases de objetos gestionados mediante GDMO [IT92b], un lenguaje de modelado de información de gestión orientado a objetos publicado por primera vez en 1992 y desarrollado de manera conjunta por ISO/IEC y el CCITT (ahora ITU-T). GDMO ha tenido una amplia repercusión en el mundo de las telecomunicaciones, tanto para el desarrollo de MIBs OSI como para la especificación de estándares en tecnologías de red. El cuadro 2.6 muestra la relación existente entre las construcciones GDMO y los conceptos generales de orientación a objetos.

GDMO es un metalenguaje de templates (ISO 10165-4) definido en base al mecanismo de macros de ASN.1 (ISO 8824). Uno de los principales inconvenientes de GDMO es su dependencia respecto del protocolo de gestión subyacente, ya que las construcciones de GDMO son específicas para CMIP [IT91a], el protocolo de gestión de OSI. A este tipo de aproximación al control se la suele denominar *API de protocolo* ya que la semántica ofrecida por el modelo de información está restringida por

Construcción GDMO	**Equivalente en OO**
MANAGED OBJECT CLASS	Clase
Instancia de MANAGED OBJECT CLASS	Objeto
ATTRIBUTE	Atributo (público) de objeto
PARAMETER	Parámetro de método
BEHAVIOR	Comentarios textuales
DERIVED FROM	Relación de herencia (múltiple)
NAME BINDING	Relación de agregación
ACTION	Método
NOTIFICATION	Método
Managed Object Boundary	Interfaz pública
Alomorfismo	Polimormismo

Cuadro 2.6: Relación entre las construcciones GDMO y los conceptos generales de orientación a objetos en que se apoyan

las primitivas del protocolo de gestión (*M-GET*, *M-SET*, *M-ACTION*). Esto supone una fuerte limitación ante la adopción de otros protocolos de gestión alternativos como SNMP o HTTP, principalmente como consecuencia de la convergencia entre las redes de telecomunicaciones y las redes de datos. Aun así, el nivel de abstracción operacional ofrecido por los objetos gestionados de OSI es claramente superior al ofrecido por las *MIBs de instrumentación* del modelo de Internet, en las que las acciones ocurrían como efecto colateral a la modificación del valor de un objeto de la MIB mediante una operación de protocolo *SNMP_SetRequest*.

Desde una perspectiva operacional, el modelo de información de OSI proporciona las siguientes posibilidades a las aplicaciones de usuario:

- Notificación de un suceso a un gestor por parte de un agente. La expresividad del modelo en este sentido es muy superior a la ofrecida por SNMPv1 (traps) y SNMPv2 (notificaciones).

- Solicitud de realización de un conjunto de operaciones de gestión a un agente por parte de un gestor. Estas operaciones incluyen el envío de la información almacenada en los atributos de los objetos gestionados mantenidos por dicho agente, la modificación de esta información y la invocación de las acciones soportadas por dichos objetos, incluida su creación/destrucción.

A nivel semántico, pese a que GDMO incluye un template para describir el comportamiento, éste se utiliza generalmente para introducir descripciones textuales. Actualmente, no existe ninguna extensión estándar al lenguaje GDMO que incorpo-

Criterio	**MIB de OSI**	**MIB de Internet**
Modelo	Orientación a objetos	Orientación a datos
Leng. de especificación	GDMO	ASN.1 (Defs. de macros)
Polimormismo	Si	No
Relac. de herencia	Si (árbol de herencia)	No
Relac. de agregación	Si (árbol de contención)	Si (árbol de registro)
Estructura de MIB	Dinámica: Árbol de nombrado (distingue clases y objetos)	Estática: Árbol de registro (no dist. clases y objetos)
Funcionalidad	Operaciones sobre MO (APIs de protocolo)	MIB de instrumentación (efectos colaterales)

Cuadro 2.7: Comparación entre las MIBs de OSI e Internet

re la utilización de alguna de las técnicas de descripción formal existentes con estos fines, como por ejemplo el lenguaje de especificación y descripción SDL (ITU-T Z.100) o Z (ISO 10165 *series*).

Otra restricción semántica de GDMO es que no permite expresar las relaciones existentes entre los objetos gestionados (salvo las relaciones de herencia). La única forma de modelar y expresar estas relaciones (de asociación, agregación, dependencia, etc.) es mediante descripciones textuales en el template *behavior*. Las aproximaciones existentes para expresar relaciones de forma indirecta consisten en utilizar punteros a objetos gestionados como tipos de datos en los atributos de otros objetos gestionados y utilizar la *ligadura a nombre* para establecer relaciones de contención (similares a las agregaciones de OO). ISO 10165-7 introduce un modelo general de relaciones que permite expresar relaciones "gestionado" como clases de objetos gestionados con operaciones *bind*, *establish*, *query*, etc. acerca de la relación. Este último modelo se ha utilizado en el entorno TMN para modelar el nivel de red.

El cuadro 2.7 recoge las principales diferencias entre las MIB de OSI e Internet.

2.2.2. Capacidades de organización

El modelo organizativo utilizado en la gestión OSI (ISO 10040) sigue el paradigma jerárquico y débilmente distribuido. Si bien el modelo ubica los sistemas en los dos roles típicos de los esquemas centralizados (gestor y agente), y por tanto sigue un modelo de cooperación asimétrico, permite que un sistema adopte ambos roles, incluso simultáneamente, de forma dinámica en función de los procesos de comunicación de gestión concretos. Esto permite crear entornos en cascada como el mostrado en la figura 2.1, en los que la interfaz local entre un agente y un gestor

en un sistema intermedio no está estandarizada, al igual que ocurre con la interfaz local entre agentes y objetos gestionados en los sistemas gestionados. Esta es una de las prácticas usuales en TMN para extender las capacidades de gestión.

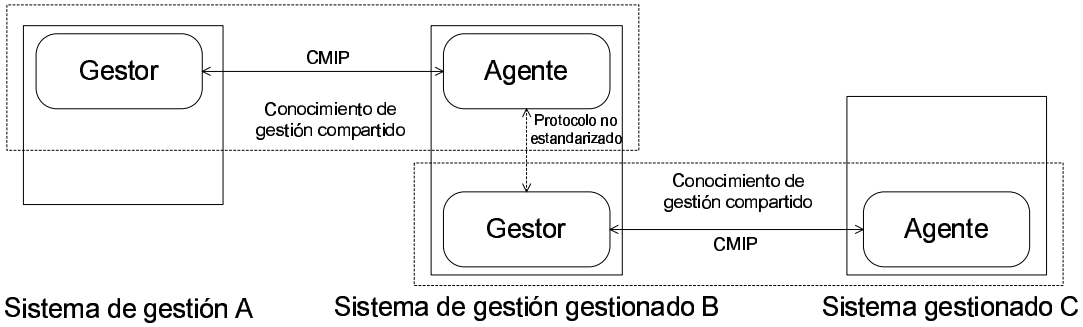

Figura 2.1: Gestores y agentes OSI en una configuración de entorno en cascada

Con el fin de permitir la cooperación entre los sistemas conectados en cascada de forma eficiente, el modelo de gestión de OSI especifica que dos sistemas deben poder compartir e interpretar sus respectivas funcionalidades de gestión en forma de *conocimiento de gestión compartido*. Este concepto incluye tanto el conocimiento de protocolo, como el conocimiento funcional y el conocimiento acerca de las clases de objetos gestionados. OSI modela este conocimiento en forma de perfiles (profiles), entre los que destacan los *international standard profiles* (ISPs) con códigos AOM1XX y AOM2XX. En este contexto, los objetos gestionados se consideran "activos" en el sentido de que son capaces de realizar notificaciones asíncronas (realmente, lo es el agente que lo implementa). Sin embargo, si se pierde laconexión con el sistema gestor, son incapaces de tomar ningún tipo de iniciativa, por lo que no pueden considerarse "proactivos".

Con el fin de superar los problemas propios del modelo centralizado en la gestión de grandes redes de comunicaciones de datos y grandes redes de telecomunicaciones, OSI introduce el concepto de *dominio de gestión*, distinguiendo entre dominios organizativos y dominios administrativos [IT92c]. Los primeros agrupan los objetos gestionados según su función, políticas (clase de procedimiento de gestión) y rol asignado. Los segundos agrupan los objetos gestionados sujetos a una misma autoridad administrativa con el fin de controlar la creación y manipulación de los dominios administrativos y el flujo de acción entre dominios solapados. Los dominios se diseñan como clases de objetos gestionados. Las políticas se especifican como clases de objetos gestionados que modelan conjuntos de reglas asignadas a un dominio con el propósito de restringir el comportamiento de sus objetos gestionados. En un mismo dominio de gestión pueden coexistir múltiples gestores y cero o más agentes. Sin

embargo, estos gestores deben estar bajo el mismo control administrativo. La figura 2.2 muestra una configuración de dominios de gestión y dominios administrativos.

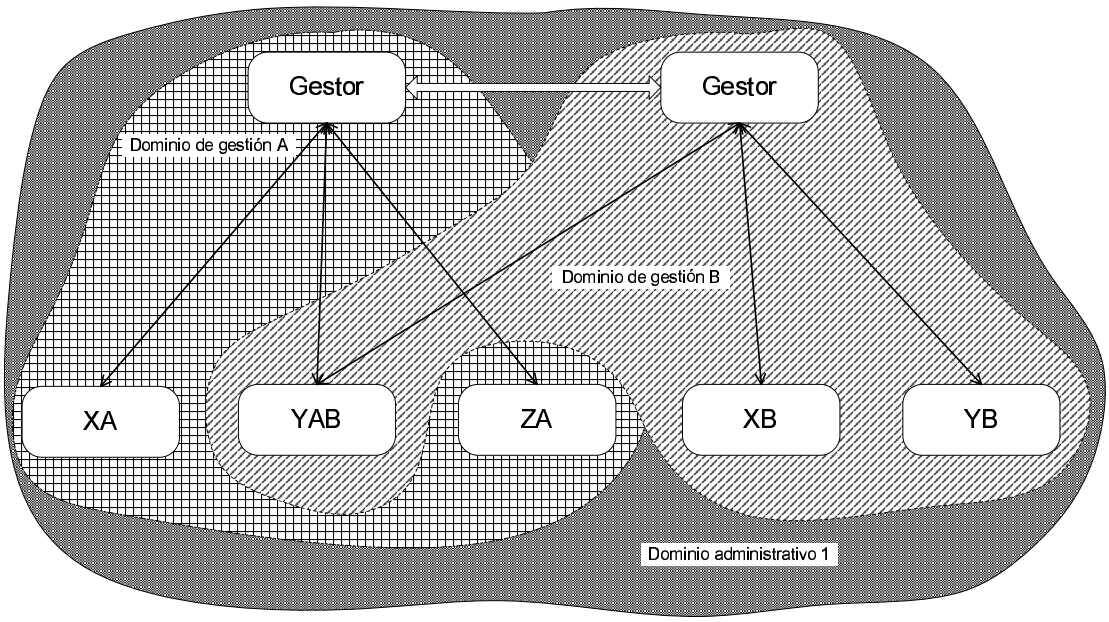

Figura 2.2: Dominios de gestión y dominios administrativos en la gestión OSI

El concepto de dominio de gestión ayuda a distribuir las funciones de gestión. Un diseño cuidadoso de los dominios de gestión permite particionar convenientemente la tarea de gestión de sistemas en subtareas más manejables y permite la aplicación de políticas de gestión a grupos de objetos al establecer un esquema de nombrado orientado a dominios en el que los dominios reciben nombres únicos. Tal y como recoge la figura 2.2, un objeto gestionado puede pertenecer simultáneamente a más de un dominio de gestión.

La tarea de gestión se complica enormemente si se considera la coordinación de gestores en diferentes dominios de gestión. Los dominios administrativos pretenden facilitar esta labor mediante:

- El control sobre los objetos gestionados y los agentes pertenecientes al dominio.
- La flexibilización de los límites de los dominios de gestión.
- la coordinación en la gestión de objetos gestionados pertenecientes a más de un dominio de gestión.

La especificación X.749 (ISO 10164-19) describe las clases de objetos gestionados para dominios de gestión y políticas de gestión, así como sus modelos de comportamiento y los servicios que pueden ser realizados. La especificación recoge también un

modelo para la obtención de información acerca de los objetos gestionados asociados a los diferentes dominios de gestión en que se haya dividido el sistema. Entre las operaciones de gestión que pueden realizarse sobre un dominio de gestión se encuentran la creación/eliminación de subdominios, la asignación/eliminación de miembros y el listado de dominios bajo las relaciones ser-subdominio-de, ser-dominio-padre-de y ser-miembro-de. La creación de un dominio por un gestor se realiza mediante la creación de una instancia de una relación de gestión entre una política de gestión, un miembro de un dominio y un coordinador de dominio.

2.2.3. Telecommunications Management Network

El marco de trabajo TMN [IT92d], desarrollado conjuntamente por el ITU-T, ETSI, ANSI, T1M1, EURESCOM, ATM Forum y el TeleManagement Forum, representa una arquitectura de aplicaciones de gestión orientada a los operadores y proveedores de redes públicas de telecomunicaciones extensamente basada en el modelo de gestión de OSI.

Todas las clases de objetos gestionados OSI y las especificaciones de templates pueden reutilizarse en un entorno TMN. El modelo de información de red genérico TMN (M.3100) define un conjunto de clases independientes de la tecnología, servicio y arquitectura TC que modelan puntos de acceso a la red, componentes de red, caminos de transmisión, etc.

El modelo de organización (actores, roles, módulos de función) viene definido en gran medida por la especificación de su interfaz X[2] y contempla la situación de los operadores en las redes públicas. La interfaz X [IT97a, IT97b] define los modos de interconexión válidos entre sistemas de operaciones ubicados en redes TMN diferentes, así como la forma de conectar una red TMN a otros sistemas de gestión de red con interfaces equivalentes. Se definen dos configuraciones diferentes a la hora de interconectar los sistemas de operaciones. El interfaz X puede utilizarse para conectar sistemas de operaciones a través de una relación igual a igual, dando lugar a un modelo de gestión de tipo cooperativo como el mostrado en la parte superior de la figura 2.3. Sin embargo, son más comunes las configuraciones establecidas en base a relaciones gestor-agente entre sistemas de operaciones, como muestra el modelo de gestión jerárquico descrito en la parte inferior de la figura 2.3. La interfaz X

[2]En TMN, el concepto de *interfaz* especifica una composición de protocolos de gestión, así como los mensajes transportados sobre dichos protocolos. Las interfaces están referidas a puntos de referencia. TMN define interfaces Q que especifican conjuntos de niveles de la pila OSI, interfaces X que especifican las formas de interconexión entre sistemas de operaciones TMN, interfaces F que especifican la interconexión de estaciones de trabajo a la información TMN, interfaces M que definen la conexión con entidades que no siguen los estándares TMN, e intefaces G que proporcionan a los usuarios el acceso a información TMN.

presenta importantes requisitos de seguridad ya que supone un punto de conexión entre diferentes sistemas TMN.

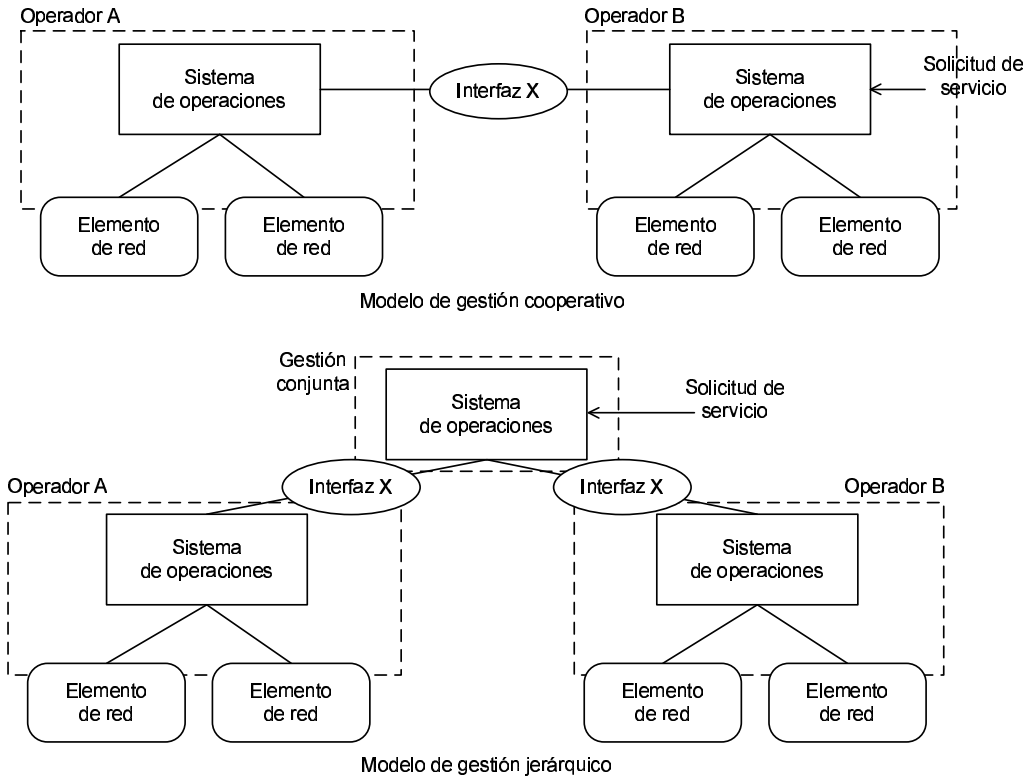

Figura 2.3: Modelos de gestión cooperativo y jerárquico definidos por la interfaz X

2.3. La arquitectura DMI del DMTF

La arquitectura DMI (Desktop Management Interface) constituye la propuesta del DMTF (Desktop Management Task Force) para la gestión integrada de sistemas finales heterogéneos tales como servidores y estaciones de trabajo, hasta entonces de difícil acceso desde las arquitecturas de gestión estándares, pese a su importancia en las infraestructuras de computación distribuidas. El principal objetivo perseguido en su diseño fue pues facilitar el acceso desde las arquitecturas de gestión existentes, principalmente la de Internet, a las interfaces propietarias ofrecidas por los diferentes sistemas operativos para la configuración de sus dispositivos.

2.3.1. Riqueza semántica de la información de gestión

El modelo de información utilizado por DMI para describir los recursos gestionados y las operaciones permitidas para su gestión sigue una aproximación a tipos

de datos muy similar a la adoptada en la SMI de Internet. La principal diferencia entre ambos modelos radica en que, además de las propiedades asociadas a un objeto gestionado, el modelo de información de la arquitectura DMI declara también su interfaz operacional mediante un lenguaje de definición de interfaces (IDL) denominado MIF (*Managed Information Format*). La descripción MIF de un componente se transfiere al SP durante su registro. En todo caso, esto no supone más que una mera MIB de instrumentación ya que, aparte de un conjunto de operaciones para administración de componentes y registro de destino de sucesos, se trata de meras operaciones de acceso y modificación de [conjuntos de] atributos a partir de las cuales deben controlarse los recursos del sistema, tal y como ocurría en el modelo de información de Internet.

La interoperabilidad de la información de gestión manejada por terceras partes se consigue mediante la estandarización de grupos de atributos predefinidos, tanto genéricos como específicos de una clase de componentes (*Systems standard group definition*, *Mass storage standard group MIF definition*, *Monitor MIF*, *Software standard group definition*, etc.). La extensión de las definiciones para sistemas finales recogidas en estos grupos es muy superior a la ofrecida por otras arquitecturas (e.g. la HOST-RESOURCES MIB del IETF).

MIF no contempla la especificación de relaciones más allá de las relaciones jerárquicas y de contención existentes entre atributos y grupos de atributos.

2.3.2. Capacidades de organización

La primera versión de DMI se publicó en 1994. La propuesta se reducía a la estandarización de la información de gestión con el fin de facilitar la gestión local de componentes fabricados por terceras partes. No es hasta la aparición de la segunda propuesta en 1996 (DMIv2) que se observa un cierto interés por distribuir la gestión, al contemplar el acceso remoto por RPC a la entidad de gestión por parte de una aplicación de gestión central e integrarlo con el protocolo de gestión SNMP.

El modelo organizativo de DMIv2 es pues muy próximo al de la arquitectura de gestión de Internet y por tanto de tipo centralizado. Las *aplicaciones de gestión* remotas solicitan información y envían órdenes a los agentes de gestión ubicados en las estaciones de trabajo (ahora denominados *proveedores de servicios* -SP) y éstos últimos les envían a su vez notificaciones asíncronas, bien mediante un mecanismo de RPC, bien mediante SNMP, y una *interfaz de gestión* (MI) estandarizada. La interfaz operacional del *proveedor de servicios* con los recursos gestionados locales (*componentes*) se denomina *interfaz de componentes* (CI) y su implementación es propietaria, pudiendo incluso no existir como tal.

La arquitectura permite la integración dinámica y coordinación de múltiples agentes en un sistema gestionado mediante la ejecución de operaciones de registro. Se trata de una propuesta similar a la propuesta de *extensibilidad de agente* formulada por el IETF (RFC 2255).

2.4. La arquitectura WBEM del DMTF

En 1996, un consorcio industrial liderado por Microsoft y CISCO Systems y denominado *WBEM initiative* lanzó la propuesta de una arquitectura abierta para la gestión basada en web de la infraestructura global de procesamiento distribuido de una organización. La propuesta se denominó inicialmente *HyperMedia Management* (HMM) y abarcaba un nuevo modelo de información denominado *HyperMedia Management Schema* (HMMS), un nuevo modelo de comunicación restringido a un nuevo protocolo de gestión denominado *HyperMedia Management Protocol* (HMMP), y un bosquejo de modelo organizativo que por el que los clientes accedían a los gestores *HyperMedia Object Manager* (HMOM).

La propuesta inicial se calificó de excesivamente revolucionaria tanto por su propuesta de un nuevo modelo de información que obligaba a que todos los sistemas utilizados hasta la fecha debiesen ser accedidos via pasarelas de gestión, como por el carácter específico de las tecnologías implicadas, que dificultaba su utilización directa en entornos Web (el protocolo de gestión HMMP, por ejemplo, no estaba basado directamente en HTTP sino en protocolos de transporte, lo cual dificultaba la utilización de navegadores en las consolas de gestión y su coexistencia con las arquitecturas de cortafuegos existentes).

Con la adopción de la iniciativa por parte del DMTF, se originó un conjunto de cambios entre los que destacaba la sustitución de HMMS por un nuevo modelo de información denominado *Common Information Model* (CIM) y su utilización conjuntamente con HTTP y XML. La utilización de HTTP como protocolo de gestión (operaciones CIM sobre HTTP) y la utilización de XML como tecnología para la representación de la información modelada con CIM (mapping CIM/XML) corrigieron el error inicial de no basar la arquitectura en tecnologías Web estándares.

2.4.1. Riqueza semántica de la información de gestión

CIM es el modelo de información desarrollado por el DMTF para representar una vista conceptual del entorno gestionado. CIM se propone unificar y extender la instrumentación y los estándares de gestión existentes utilizando construcciones y técnicas de diseño orientadas a objetos.

El valor de CIM proviene en gran medida de su carácter orientado a objetos. Las técnicas de modelado orientado a objetos subyacentes en el meta-esquema CIM y en su sintaxis de representación textual MOF le aportan un poder expresivo netamente superior a otros formatos de información "planos" como la SMI de Internet, e incluso orientados a objetos como GDMO de OSI (dependiente del protocolo de gestión subyacente).

La capacidad de abstracción y clasificación de CIM le permite definir los conceptos fundamentales del dominio de gestión (objetos), agrupar éstos en tipos (clases) identificando sus características comunes (propiedades), relaciones (asociaciones) y comportamiento (métodos). La clasificación en subtipos permite utilizar el nivel de detalle y complejidad adecuado en cada lugar del modelo.

Antes de CIM, los estándares de gestión capturaban la semántica de las relaciones existentes entre los objetos gestionados mediante arrays multidimensionales o tablas de referencias cruzadas. El paradigma de objetos ofrece una aproximación más elegante al modelar directamente las relaciones y asociaciones. El modo en que se nombran y definen estas relaciones (asociaciones, agregaciones, dependencias) describe formalmente su semántica. Su presencia en el modelo de información ayuda a separar la información perteneciente a los objetos gestionados de la información específica de las relaciones entre éstos.

El meta-esquema CIM es un metamodelo[3] orientado a objetos y fundamentado en el lenguaje de modelado UML, que define formalmente los elementos utilizados para expresar el modelo CIM, su utilización y su semántica. Los elementos del meta-esquema son *esquemas*, *clases* (tipos), *propiedades* (estado), *métodos* (comportamiento), *calificadores* (metainformación) y disparadores. El meta-esquema también contempla *indicaciones* y *asociaciones* como tipos de clases, *referencias*(roles desempeñados en una asociación) como tipos de propiedades. Las clases CIM se organizan en jerarquías de generalización (subtipado) mediante la relación de herencia única. Las propiedades y los métodos tienen relaciones reflexivas de sobreescritura. La figura 2.4 muestra los diferentes elementos del meta-esquema CIM mediante un diagrama de clases UML.

La notación del meta-esquema CIM está basada directamente en la notación UML del OMG, por lo que permite modelar gráficamente la información de gestión sin menoscabo de su carácter formal.

La información de gestión modelada gráficamente mediante la notación del meta-

[3]La terminología de *esquemas* se utiliza comúnmente en el área de las bases de datos, mientras que la terminología de *modelos* es más propia del área de la orientación a objetos, especialmente en el entorno OMG. IETF, ITU-T e ISO se decantan también por el uso de *modelo* frente a *esquema*. El DMTF heredó la terminología de *esquemas* utilizada en la propuesta inicial del HMMS, si bien con su evolución hacia la orientación a objetos utiliza ambas terminologías como sinónimas.

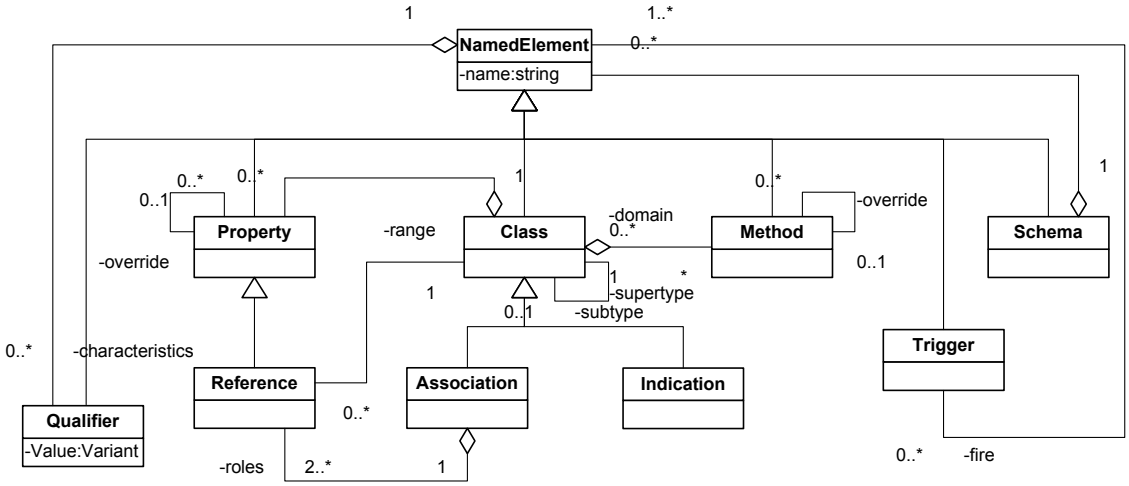

Figura 2.4: Elementos del meta-esquema CIM

esquema CIM se describe sintácticamente en un lenguaje de especificación denominado MOF (*Managed Object Format*) basado en el lenguaje de definición de interfaces IDL. La sintaxis MOF es un modo de describir las definiciones de objetos gestionados basadas en el meta-esquema CIM en formato textual. Un servidor puede validar la corrección de las descripciones de objetos gestionados incluidas una especificación MOF tanto sintácticamente, como semánticamente contra una implementación concreta. La notación MOF se basa en un lenguaje de plantillas semejante al utilizado por ISO en la notación GDMO o al utilizado por el IETF en para la SMI.

MOF especifica la semántica de los atributos CIM de manera informal mediante descripciones textuales y de manera formal a través de construcciones sintácticas para los calificadores ofrecidos por el meta-esquema CIM. Pueden distinguirse dos tipos de calificadores:

Calificadores de tipos de datos CIM : Algunos tipos de datos intrínsecos se suelen utilizar con semánticas bien definidas (e.g. octetstring, counter, gauge, ArrayType("Indexed")) originando así pseudo-tipos (ya que no se tratan como tipos intrínsecos). En CIM no se asocia ninguna semántica formal a estos calificadores. Diferentes implementaciones pueden introducir calificadores arbitrarios que tan solo se contemplan con propósitos de documentación. De hecho, la especificación del modelo CIM no incluye ninguna relación de calificadores estándar.

Calificadores de NamedElement : Califican clases, asociaciones, indicaciones, métodos, parámetros, triggers, instancias, propiedades y referencias. Se utilizan para calificar estos elementos nombrados. Proporcionan un mecanismo

Especificación	**Versión**	**Documento**
Unified Modeling Language (UML)	1.5	formal/03-03-01
Meta-Object Facility (MOF)	1.4	formal/02-04-03
XML Metadata Interchange (XMI)	2.0	formal/03-05-02
CORBA/TMN Internetworking	1.0	formal/00-08-01
CORBA/TC-IN Internetworking	1.0	formal/01-01-01
CORBA/IIOP Core Specification	3.0.2	formal/02-12-02

Cuadro 2.8: Estándares del OMG afines a la gestión de sistemas distribuidos

de extensibilidad controlada al meta-modelo CIM, ya que añaden información semántica adicional a estos elementos. Todos los calificadores constan de un nombre, un tipo, un valor, un alcance, un comportamiento y un valor por omisión. Algunos de estos calificadores son mutuamente exclusivos y el uso de alguno de ellos implica algunas restricciones en el valor de otro calificador.

En el año 1998, el DMTF comenzó a desarrollar una especificación para la representación estándar en XML de sus modelos de información CIM, basada hasta entonces en MOF (Managed Object Format). Surge así la especificación del mapping CIM-XML, también denominado *CIM-XML Schema*. La representación XML de CIM constituye un paso adelante en la especificación transferible e interpretable de forma automática de metainformación asociada a los modelos CIM.

2.5. OMA como arquitectura de gestión

La arquitectura de gestión de objetos OMA (del inglés *Object Management Architecture*) desarrollada por el OMG (*Object Management Group*) no ha sido concebida específicamente para la gestión de infraestructuras de comunicaciones sino como una arquitectura de propósito general para el desarrollo sistemas distribuidos. Sin embargo, está recibiendo creciente atención en el área de la gestión integrada de redes y sistemas, debido fundamentalmente al interés prioritario por disponer de una arquitectura común para desarrollo, implantación, utilización y gestión de los sistemas distribuidos, las infraestructuras de red que los soportan y los servicios ofrecidos en base a los mismos. El objetivo perseguido es utilizar las mismas herramientas para el desarrollo y la gestión de los sistemas distribuidos y OMA se presenta como una de las propuestas más prometedoras por su afinidad con otros estándares del OMG como UML, CORBA-IDL, MOF o XMI. El cuadro 2.8 recoge el estado actual de las especificaciones de éstos estándares.

OMA proporciona un marco de trabajo para el desarrollo de sistemas distri-

buidos orientados a objetos con independencia del lenguaje de programación y la plataforma de ejecución utilizados que facilita la cooperación de objetos en un entorno heterogéneo con transparencia de localización e implementación.

Como parte del marco de trabajo proporcionado por OMA se han definido diversas subarquitecturas y especificaciones. La arquitectura para intermediación de peticiones entre objetos comunes CORBA (del inglés *Common Object Request Broker Architecture* constituye el núcleo de OMA. CORBA define principalmente el modelo de comunicación de OMA al especificar las relaciones cliente-servidor que rigen las interacciones entre los objetos distribuidos a través de un intermediario de peticiones denominado ORB. Pero no sólo define su modelo de comunicación, también define su modelo de información a través de su modelo de objetos y su lenguaje de definición de interfaces (IDL), su modelo organizativo a través de su arquitectura de interoperabilidad y su modelo funcional a través de sus servicios y facilidades comunes de objetos.

2.5.1. Riqueza semántica de la información de gestión

El modelo de información de OMA está totalmente orientado a objetos y se basa en el lenguaje estándar de modelado UML (del inglés *Unified Modelling Language*), un lenguaje gráfico orientado a la visualización, especificación, construcción y documentación de los artefactos de un sistema software. UML puede utilizarse tanto para el modelado estructural (diagramas de clases y diagramas de objetos) como el modelado de comportamiento (colaboraciones, casos de uso, máquinas de estados y diagramas de actividad) y su notación gráfica está respaldada por una sólida formalización semántica. La notación UML cubre las diferentes fases del ciclo de vida del software (modelado del negocio, captura de requisitos, análisis, diseño, construcción y despliegue).

Dentro de OMA, la arquitectura CORBA especifica un modelo de objetos y una sintaxis para la descripción formal de las interfaces de objetos. Una de las principales ventajas del modelo de información OMA frente a otros modelos como el de Internet o el de OSI es que resulta menos específico y totalmente independiente del protocolo de comunicación subyacente. Esto fomenta la creación de modelos portables. El lenguaje de especificación de las interfaces de objetos (IDL, del inglés *Interface Definition Language*) es a su vez independiente del lenguaje de programación utilizado en su implementación lo cual favorece aún más la interoperabilidad.

El modelo de objetos de CORBA introduce la base conceptual para la extensión y refinamiento en forma de componentes. El modelo de componentes de CORBA especifica un marco de trabajo para el desarrollo de componentes que encapsulan su

creación, la gestión de su ciclo de vida y sus sucesos y que facilitan su exploración (reflexión).

El modelo de objetos de OMA, debido fundamentalmente a su carácter general, no incorpora ningún concepto similar a los sucesos (traps) del modelo de Internet o las notificaciones del modelo OSI. Esta funcionalidad se contempla como un conjunto de servicios comunes dentro del modelo funcional de OMA tanto en su versión pull como push (servicio de sucesos, servicio de notificación y servicio de notificación tipada).

El lenguaje de especificación IDL permite utilizar técnicas de orientación a objetos en la especificación formal de las interfaces de los objetos gestionados y separa dicha especificación de los detalles de implementación, con los consiguientes beneficios en cuanto a encapsulación, neutralidad de implementación e independencia de la plataforma. Sin embargo, no es una buena herramienta de modelización a la hora de tratar relaciones estructurales complejas o comportamiento complejo. A pesar de que la utilización de UML permite capturar gráficamente la semántica de las relaciones existentes entre los objetos descritos, el lenguaje IDL no permite expresar estas relaciones, más allá de las relaciones de herencia, si bien determinadas relaciones aparecen reflejadas a través de servicios comunes tales como el servicio de vida o el servicio de relaciones.

OMA facilita la especificación formal y la comunicación de los modelos de información desarrollados en UML a través de sus especificaciones MOF (Meta-Object Facility) y XMI (Xml Metadata Interchange). MOF y XMI automatizan la implementación de repositorios de metainformación con formatos intercambiables comunes para el dominio de gestión. Esta integración de los estándares de metainformación y modelado del OMG y del W3C facilita el modelado, descubrimiento, gestión y compartición de metainformación a través de Internet.

En noviembre de 1997, el *Object Analysis and Design Task Force* del OMG desarrolló la facilidad MOF para la definición, descubrimiento e integración de metainformación en repositorios estándar. El metamodelado basado en MOF constituye una técnica formal para diseñar e implementar tanto lenguajes de modelado (como por ejemplo UML), como repositorios de metadatos destinados a integrar aplicaciones y herramientas en entornos distribuidos (como el *Common Warehouse Metamodel* -CWM). El metamodelado proporciona los fundamentos para la introspección y reflexión no solo de atributos y operaciones, sino también de relaciones, restricciones, etc.

Si bien MOF ya ofrecía la posibilidad de intercambio de metainformación a través de interfaces CORBA mediante la *CORBA UML Facility*, la fuerte aceptación

Metamodelo UML	**Meta-metamodelo MOF**	**Mod. de objetos OMA**	**IDL de CORBA**
Association (n-aria)	Association (binaria)		
AssociationClass	NA		
AssociationEnd	AssociationEnd		
Attribute	Attribute	Attribute	Attribute
BehavioralFeature	BehavioralFeature		
Class	Class	Class	Interface (como Class)
Classifier	Classifier		
Constraint	Constraint		
Datatype	Datatype	Tipo de datos	Tipo de datos
Dependency (clase)	/dependsOn (asociación)		
Exception	Exception		Exception
Feature	Feature		
GeneralizableElement	GeneralizableElement		
Generalization (clase)	generalizes (asociación)	Generalization	Generalization
Interface	Class (como Interface)	Interface	Interface
ModelElement	ModelElement		
NA	Reference		
NA	Constant		Constant
Namespace	Namespace		$\sim$ Containers (IR)
Operation	Operation	Operation	Operation
Package	Package		Module
Parameter	Parameter	Parameter	Parameter
StructuralFeature	StructuralFeature		
Type (estereotipo)	Class (como Type)	Type	Interface (como Type)

Cuadro 2.9: Comparación de la expresividad de los modelos OMA

de XML en el área del modelado de información provocó que en marzo de 1999 el OMG estandarizase XMI (XML Metadata Interchange), una especificación utilizada para el intercambio XML de metainformación entre herramientas, repositorios y aplicaciones compatibles con MOF como conjuntos de información XML (XML *datasets*), con independencia de la plataforma [4]. XMI permite el intercambio tanto de modelos de aplicación UML como de modelos de datos CWM, por lo que puede verse como un "mapping" de estos lenguajes de modelado a XML.

El cuadro 2.9 compara los principales conceptos de modelado expresables en UML, en MOF, en el modelo de objetos de CORBA y en el IDL. En ella se aprecia la escasa expresividad del modelo de objetos de OMA y del IDL de CORBA con respecto a la riqueza semántica del metamodelo UML, así como la adecuación de MOF y XMI para el intercambio de modelos UML entre entidades software.

El mismo modelo de objetos y el mismo lenguaje de especificación que conforman

[4]Tanto MOF como XMI representan un marco de integración, independiente de la plataforma y guiado por los modelos, para el intercambio, manejo e integración de [meta]información. XMI utiliza tecnología XML para guiar la generación de esquemas XML para cualquier [meta]modelo basado en MOF.

el modelo de información de OMA sirve para definir las interfaces de los objetos de nivel superior y que constituyen el modelo funcional de OMA. Estas interfaces se agrupan en (1) interfaces de servicios comunes independientes del dominio e (2) interfaces de facilidades comunes tanto independientes del dominio (verticales) como específicas de un dominio (horizontales). Entre estas últimas destacan las facilidades comunes para gestión que especifican los servicios de gestión integrados en OMA. Las aplicaciones de gestión tienen la posibilidad de usar todos los servicios definidos en los diferentes niveles de este modelo funcional estratificado.

2.5.2. Capacidades de organización

El modelo organizativo de OMA sigue el paradigma fuertemente distribuido y facilita un modelo de comunicación cooperativo simétrico entre objetos igual a igual (peer-to-peer). El modelo es transparente a la localización e independiente de la plataforma gracias a los, así denominados, protocolos generales inter-ORB (GIOP). La arquitectura de interoperabilidad de CORBA, basada en el concepto de dominio, es el componente principal de dicho modelo organizativo.

El hecho de que se trate de un modelo organizativo fuertemente distribuido y cooperativo y no de un modelo estrictamente jerárquico añade la posibilidad de crear relaciones agente-agente además de las típicas relaciones agente-gestor o gestor-gestor como en OSI e Internet. Esto no sólo permite distribuir la funcionalidad de gestión entre sistemas jerárquicamente iguales y delegar los servicios de gestión a sistemas jerárquicamente inferiores, sino que en principio también permite la cooperación autónoma entre sistemas con rol de agente. El modelo contempla exclusivamente objetos que, en función de los requisitos o de la implementación, pueden asumir el rol de un cliente, de un servidor, o ambos simultáneamente.

Este modelo de organización inicial se extendió considerablemente en la versión 2 de CORBA para incluir una estructura de interoperabilidad que solventase las dificultades de interoperabilidad existentes entre ORBs de diferentes proveedores. La interoperabilidad técnica de entre ORBs aportada por la versión 2 constituyó la base para la federación de ORBs independientes a través de límites tanto organizativos como tecnológicos. La existencia de dichos límites pasa a ser totalmente transparente. El prerrequisito para llevar a cabo este tipo de federación es el establecimiento de dominios y pasarelas entre diferentes dominios.

Un dominio en el modelo de organización de OMA es un conjunto de objetos (los miembros del dominio), con ciertas propiedades en común. Los dominios se modelan a su vez como objetos, de modo que puedan ser a su vez miembros de otros dominios. Estos dominios, tal y como ocurría con los dominios de la arquitectura de gestión

OSI de ISO, se dividen en administrativos y técnicos.

El concepto de dominio y la posibilidad de federar los ORBs permiten la comunicación a través de fronteras tanto organizativas como tecnológicas.

2.5.3. La arquitectura TINA

La complejidad inherente a la arquitectura OMA y a su pieza fundamental: CORBA, ha orientado su utilización como arquitectura de gestión de redes de telecomunicaciones. TINA y TMN constituyen los dos principales ejemplos de adopción de CORBA como arquitectura para sistemas distribuidos subyacente.

La arquitectura TINA (*Telecommunications Information Networking Architecture*) constituye realmente un marco de trabajo y una infraestructura para el desarrollo de aplicaciones distribuidas en el contexto de las redes de telecomunicaciones. Más aún, ha sido diseñada con el objetivo de integrar otras arquitecturas de telecomunicaciones como la Red Inteligente (IN) [5] y TMN. La arquitectura aplica los principios de RM-ODP y la tecnología CORBA a los requisitos específicos del dominio de las telecomunicaciones. Así, su infraestructura/plataforma de despliegue y ejecución de aplicaciones distribuida, denominada entorno de procesamiento distribuido (DPE, del inglés *Distributed Processing Environment*) ha sido especificada utilizando el modelo computacional y el modelo de ingeniería de RM-ODP y está basada ampliamente en la tecnología ORB de CORBA, como demuestra el hecho de que el lenguaje de definición de objetos (ODL, del inglés *Object Definition Language*) propuesto en su modelo de información sea una mera extensión del IDL de CORBA.

2.6. Gestión basada en políticas PBN

La gestión basada en políticas (PBN, del inglés *Policy-Based Networking*) [Ver01] representa un cambio importante en la forma en que se ha venido concibiendo tradicionalmente la gestión de red. En lugar de concentrarse en los dispositivos e interfaces de red o en los propios sistemas distribuidos, centra su atención en los usuarios y los servicios ofrecidos. Con este fin, el modelo de gestión PBN se propone capturar y definir formalmente las reglas de negocio en forma de requisitos al más alto nivel y establecer un proceso determinista de traducción de estos requisitos a políticas es-

[5] La idea subyacente en el concepto de Red Inteligente (IN, del inglés *Intelligent Network*) es la provisión de facilidades para que un operador de red o un proveedor de servicios pueda definir nuevos servicios (reenvío de llamada, llamada a tres, anulación de prefijos, etc.) a partir de la integración de componentes de servicio reutilizables y de un conjunto definido de capacidades. La lógica del nuevo servicio se construye a partir de software de funcionalidad específica, y se proporciona de manera centralizada desde nodos especializados denominados SCP (*Service Control Points*).

pecíficas que ligen las necesidades del negocio con un comportamiento coherente de la red y los sistemas distribuidos, definido a alto nivel. El sistema de gestión basado en políticas se encargará entonces del proceso de transformación de éstas políticas en representaciones operacionales aptas para ser interpretadas por los agentes de gestión que operan en los dispositivos de red y cuyos modelos de información pueden corresponder a cualquiera de las arquitecturas de gestión vistas. Por tanto, un sistema de gestión basado en políticas no constituye en sí mismo una arquitectura y sí un marco de trabajo que debe ser integrado en los sistemas de gestión utilizados de forma regular. Aun así, es posible hablar de PBN en función de un modelo de información y un modelo organizativo.

El modelo de información más conocido es el Directory Enabled Networks (DEN), que ya forma parte del CIM. DEN modela las políticas de un sistema de gestión desde una aproximación orientada a directorios. El modelo debe proporcionar además un lenguaje de especificación de políticas que permita representar los requisitos y las funciones de negocio con independencia del dispositivo final. El lenguaje debe por tanto permitir abstraer las especificaciones de las políticas de los comandos de configuración específicos del elemento gestionado, así como asegurar la consistencia de las políticas. Este último punto conlleva la necesidad de ofrecer un mecanismo de resolución de conflictos. El lenguaje de especificación es el elemento del modelo que más atención está recibiendo últimamente [Dam02].

El modelo organizativo de PBN proporciona un marco para la administración, gestión y distribución de las políticas en el sistema de gestión. El IETF está trabajando actualmente en un estándar para la traducción de los esquemas CIM relacionados con DEN en LDAP con el fin de facilitar la administración y gestión de directorios de políticas y ha desarrollado un protocolo para distribución de políticas denominado COPS (*Common Open Policy Service*). El modelo de cooperación entre las diferentes entidades participantes (PDPs, PEPs y agentes) es de tipo jerárquico y se basa en delegación por dominios. La información de políticas se almacena en un conjunto de bases de políticas (MPBs) organizadas según dominios. Cada agente PDP extrae las políticas almacenadas en la base de política (MPB) de la que es responsable y efectúa un proceso de toma de decisiones. Los PDPs envían entonces estas decisiones, expresadas aún con independencia del dispositivo a los PEP asociados, que se encargan de traducirlas en operaciones o comandos específicos para una tecnología concreta y según modelo de información de los agentes que actúan en los recursos gestionados por dichos PEPs.

2.7. Análisis de las arquitecturas existentes

Las figuras 2.5 y 2.6 ofrecen una comparativa a nivel cualitativo entre los modelos de información de las diferentes arquitecturas de gestión existentes desde las perspectivas de su nivel de abstracción y su capacidad expresiva. Ambas figuras muestran también la ubicación de la arquitectura propuesta (Nesmarq) respecto de estos parámetros de clasificación.

La figura 2.5 muestra cómo la arquitectura de gestión SNMP es la que ofrece un menor nivel de abstracción al estar basada en una SMI orientada a datos sin capacidad para describir operaciones. Como ya se ha comentado, estas últimas se producen como efecto colateral al modificar el valor almacenado por los objetos de la MIB a través de las operaciones ofrecidas por el protocolo de gestión. Se habla entonces de *MIBs de instrumentación*. La arquitectura DMIv2 del DMTF ofrece un nivel de abstracción similar, si bien permite describir las operaciones de lectura/escritura sobre atributos y grupos de atributos y operaciones de registro de componentes mediante un lenguaje de tipo IDL.

Por su parte, la arquitectura de gestión de OSI (y por tanto la propuesta TMN) ofrece mayor riqueza semántica que las anteriores al estar basada en una SMI orientada a objetos. Sin embargo, su modelo de información aún está estrechamente ligado al protocolo de gestión. Se habla entonces de *APIs de protocolo*.

Las arquitecturas basadas en tecnologías de objetos distribuidos como JMAPI (basada en el modelo de objetos de Java), WBEM (basada en CIM) u OMA (basada en el modelo de objetos de CORBA) ofrecen un mayor nivel de abstracción gracias a la utilización de lenguajes IDL independientes del protocolo de gestión subyacente. Se habla entonces de *objetos computacionales*. OMA ofrece mayor expresividad que el resto de arquitecturas basadas en objetos computacionales, gracias a la utilización del lenguaje de modelado UML, que permite el modelado visual de relaciones tanto estructurales como de comportamiento, y los estándares MOF y XMI para especificación y comunicación de metainformación. El modelo CIM del WBEM ofrece una expresividad semejante al estar basado en un meta-esquema provisto de una notación gráfica similar a la propuesta por el OMG para UML, y disponer también de un *mapping* CIM/XML que facilita la especificación y el transporte de metainformación. La principal ventaja del modelo CIM frente a OMA es que al tratarse de un modelo de información específico para gestión, se dispone de un gran volumen de modelos ya elaborados y estandarizados por el propio DMTF. Por contra, su desventaja fundamental es que el modelo no tiene soporte para la especificación de comportamiento. En este sentido, el modelo de información asociado a la arquitectura JMAPI no aporta capacidades de modelado propias si bien, al tratarse de un

modelo orientado a objetos, nada le impide utilizar los estándares de OMA (además, la disponibilidad de un estándar de interoperabilidad JRMP-IIOP facilita la utilización de todos los servicios y facilidades de CORBA). Lo mismo sucede con la arquitectura TINA.

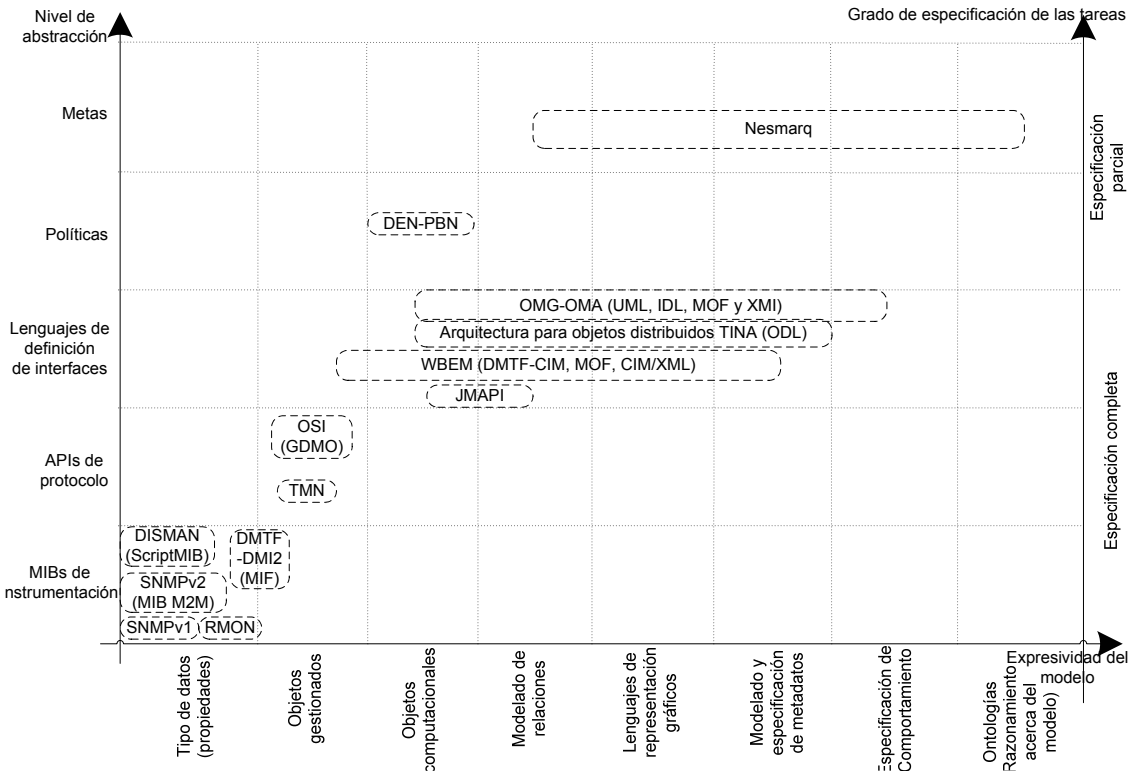

Figura 2.5: Comparativa entre los modelos de información de las arquitecturas de gestión existentes y la propuesta (*Nesmarq*)

A la vista de la figura 2.5, se puede decir que todas las arquitecturas propuestas requieren de la especificación completa de las tareas de gestión a realizar salvo la propuesta de gestión basada en políticas DEN-PBN que tan solo requiere la especificación parcial de tareas en forma de objetivos.

El análisis de las arquitecturas existentes desde la perspectiva única del modelo de información resulta insuficiente a la hora de clasificar dichas arquitecturas (la figura 2.5 muestra las arquitecturas agrupadas en tres grandes bloques). Sin embargo, si introducimos como criterio de análisis el tipo de delegación permitido por sus respectivos modelos de organización y soportado por sus correspondientes modelos de información, se observa una mejor discriminación, tal y como muestra la figura 2.6. La explicación de esta figura se corresponde en gran medida con la realizada a continuación para la figura 2.7.

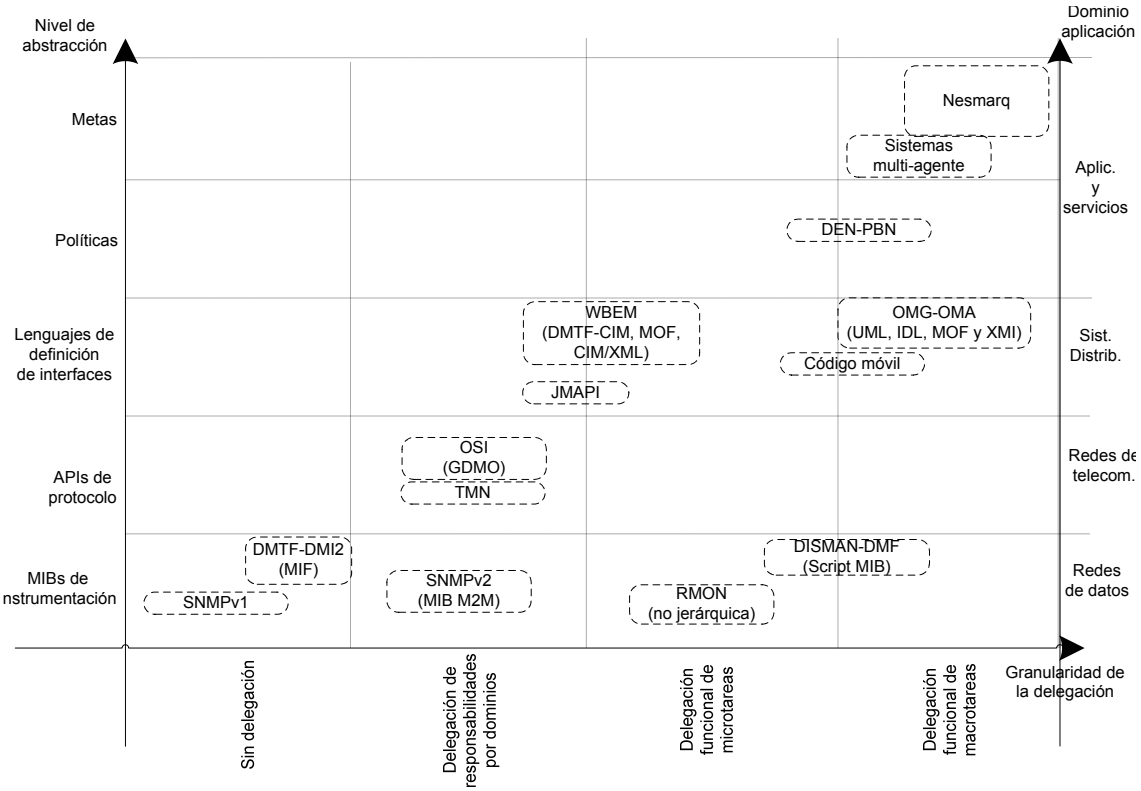

Figura 2.6: Comparativa entre los modelos de información de las arquitecturas de gestión existentes y la propuesta (*Nesmarq*)

La figura 2.7 se centra en el modelo organizativo y ofrece una comparativa entre las arquitecturas de gestión existentes desde las perspectivas de la forma en que se relacionan y cooperan las entidades del sistema y como se lleva a cabo la distribución de la funcionalidad de gestión. Obviamente, la ausencia de delegación implica ausencia de distribución y por tanto un modelo organizativo de tipo centralizado, y viceversa, por lo que ciertas regiones del gráfico son inconsistentes y aparecen marcadas como tales.

Si bien la figura 2.7 destaca cómo arquitectura de gestión de Internet y el modelo DMIv2 conllevan generalmente un modelo organizativo de tipo centralizado en el que toda la funcionalidad de gestión se concentra en el gestor y los agentes actúan como meros recolectores de información, dicha figura muestra también la evolución que se ha producido en éste modelo hacia uno más flexible de tipo jerárquico débilmente distribuido con la aparición primero de las MIBs RMON y RMON2 y con la posterior estandarización de la MIB M2M (*Manager to Manager*) en la arquitectura SNMPv2. Con RMON se permite la distribución de una pequeña parte de la funcionalidad del gestor entre los agentes de gestión (tan solo se delega en éstos tareas

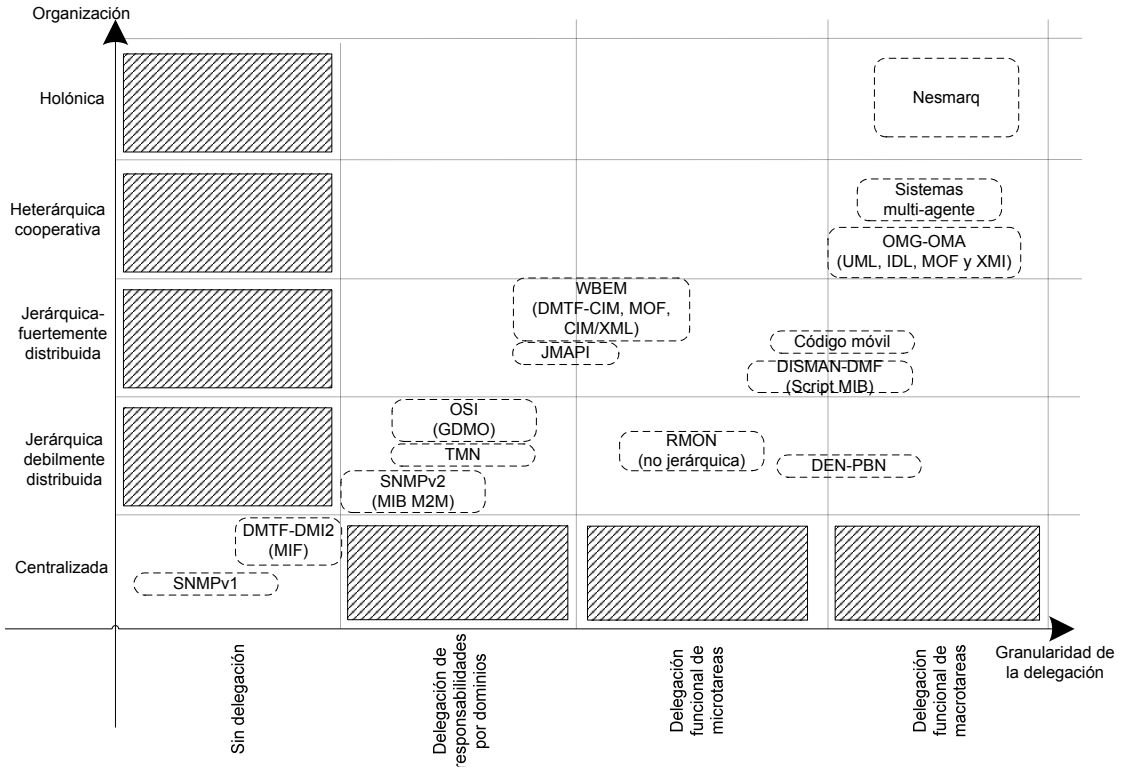

Figura 2.7: Comparativa entre los modelos de organización de las arquitecturas de gestión existentes y la propuesta (*Nesmarq*)

de agregación, preparación, clasificación y análisis estadístico de datos) y no se contempla la delegación de responsabilidades por dominios. Por su parte, la utilización de la MIB M2M y la operación de protocolo *Report* en SNMPv2 permite una distribución jerárquica de la responsabilidad de gestión por dominios. Cada dominio está representado por un gestor diferente que en principio asume toda la funcionalidad de gestión, por lo que no contempla delegación de tareas. En ambos casos se puede hablar de una distribución débil y jerárquica.

Con la posterior propuesta de distribución DISMAN-DMF, viable gracias al modelo de seguridad de SNMPv3, se evoluciona hacia un modelo distribuido en el que se permite la verdadera delegación de funcionalidad a los agentes en forma de *scripts*. La magnitud de las tareas delegadas y las facilidades de planificación ofrecidas aumentan considerablemente la escalabilidad y la robustez del sistema, por lo que puede empezar a hablarse de un modelo jerárquico fuertemente distribuido.

La figura 2.7 muestra también como la gestión OSI se basa en un modelo jerárquico débilmente distribuido en el que la distribución se produce en forma de delegación por dominios de la responsabilidad de gestión entre una red (general-

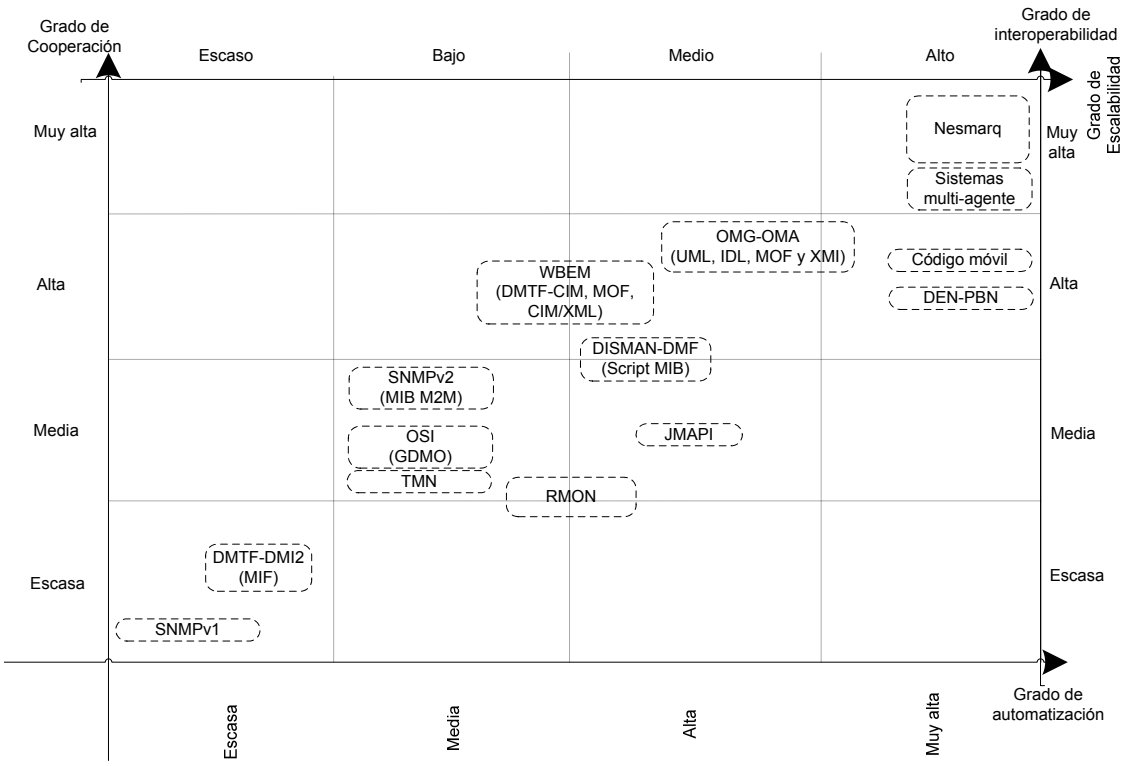

Figura 2.8: Comparativa a nivel cualitativo entre de las arquitecturas de gestión existentes y la propuesta (*Nesmarq*)

mente jerárquica) de gestores y no en forma de delegación de tareas a los agentes. El control ejercido por los gestores sobre los objetos gestionados y los agentes pertenecientes a su dominio de gestión y la coordinación de los objetos gestionados y los agentes pertenecientes además a otros dominios de gestión se realiza en base al concepto de dominio administrativo. Siguiendo las directrices de la arquitectura OSI, TMN define a partir del concepto de dominio dos modos de interconexión válidos entre sistemas de operaciones ubicados en redes TMN diferentes. Por una parte, permite la interconexión a través de relaciones *igual a igual*, dando lugar a un modelo de gestión de tipo cooperativo. Sin embargo, son más comunes las configuraciones establecidas en base a relaciones gestor-agente entre sistemas de operaciones, por lo que la figura clasifica TMN como un modelo de gestión jerárquico débilmente distribuido.

El modelo jerárquico fuertemente distribuido se afianza con las arquitecturas de gestión basadas en arquitecturas de objetos distribuidos como OMA-CORBA o JMAPI. El modelo de cooperación típico en estas últimas es de tipo *igual a igual*, lo que facilita la distribución flexible y a gran escala de las tareas de gestión entre los

diferentes objetos cooperantes. Desde la aparición con CORBA 2 de una estructura de interoperabilidad orientada a la federación de ORBs independientes a través de límites tanto organizativos como tecnológicos, nada impide en OMA la delegación de responsabilidades por dominios.

Del análisis efectuado se desprende que *no existe ninguna arquitectura idónea para todo tipo de gestión*. Cada una ofrece diferentes herramientas de modelado y utiliza diferentes tecnologías para cubrir diferentes necesidades. Así, la arquitectura de gestión de Internet resulta adecuada para el dominio de gestión de redes de datos basadas en TCP/IP, en el cual la sencillez de sus modelos ha resultado un factor determinante para su éxito, en detrimento de otras aproximaciones más elaboradas como la de OSI. Es más, los problemas de escalabilidad derivados de la aproximación centralizada seguida en su modelo organizativo y de la excesiva utilización de sondeos reducen su alcance a la gestión de redes de área local y los problemas de seguridad arrastrados hasta la versión 3 restringen su funcionalidad a labores de monitorización. Las tecnologías ofrecidas son específicas del dominio de gestión y por tanto requieren de formación específica.

La arquitectura de gestión de OSI sin embargo ha tenido mejor acogida en el mundo de las telecomunicaciones, principalmente a partir de su adopción por ITU-T en 1992 como base para el modelo TMN. TMN/OSI distribuye la gestión a lo largo de una jerarquía de gestores encargados de diferentes dominios de gestión. Lo mismo ha ocurrido con la arquitectura TINA. Si bien SNMP ha ido evolucionado también hacia un modelo jerárquico débilmente distribuido con sus propuestas RMON, SNMPv2 + MIB M2M y principalmente con DISMAN, predomina su utilización bajo una perspectiva centralizada. En ambos casos se trata de una delegación dirigida y por tanto de tipo reactiva.

A partir de la información recogida en las figuras 2.5, 2.6 y 2.7 se puede elaborar una clasificación de las arquitecturas de gestión estandarizadas como la mostrada en la figura 2.8 y que atiende los siguientes criterios:

- Grado de automatización.
- Grado de interoperabilidad.
- Grado de cooperación.
- Grado de escalabilidad.

La figura 2.8 muestra como la creciente complejidad de las redes de comunicaciones y los servicios ofrecidos sobre las mismas requiere de un grado cada vez

mayor de automatización en su gestión. Con la automatización se persigue una mayor escalabilidad y una mayor robustez de las soluciones de gestión. El grado de automatización de una solución de gestión está directamente relacionado con el paradigma seguido por el modelo organizativo de la arquitectura en que está basada dicha solución, y más concretamente, con el tipo de delegación ofrecida.

Por otra parte, para conseguir un mayor grado de automatización, también se requiere que el modelo de información permita tanto a los agentes como a los gestores realizar procesos de razonamiento automático acerca del conocimiento de gestión que comparten y que utilizan en sus interacciones. Se hace por tanto imprescindible también dotar al modelo de información de elementos que permitan un mayor grado de interoperabilidad.

Por último, el grado de cooperación alcanzado por una solución de gestión es consecuencia directa del tipo de modelo de organización soportado por la arquitectura subyacente y del modelo de interacción. Desde el punto de vista organizativo, los modelos jerárquicos fuertemente distribuidos y los modelos cooperativos son los más flexibles a la hora de distribuir la gestión. Desde el punto de vista del modelo de interacción, se hace imprescindible proporcionar soporte para la especificación de protocolos de interacción que estructuren convenientemente los procesos de negociación y cooperación que se deseen automatizar.

La figura 2.8 muestra también una marcada tendencia de las arquitecturas de gestión existentes por desarrollar modelos que permitan un alto grado de automatización en la gestión, con el consiguiente incremento del grado de escalabilidad. La figura muestra también cómo esta tendencia implica la necesidad de proveer mediante estos modelos de un mayor grado de cooperación e interoperabilidad. Todo ello conduce a la necesidad de introducir nuevas tecnologías en el desarrollo de los modelos de la arquitectura que faciliten la consecución de estos objetivos. Entre estas tecnologías podemos destacar los agentes autónomos y sistemas multi-agente, los sistemas formales de representación del conocimiento y los lenguajes de ontologías.

El éxito en el proceso de incorporación de agentes autónomos racionales, capaces de razonar e integrar de forma dinámica conocimiento y servicios, depende en gran medida de la evolución de los modelos de información de las arquitecturas de gestión a modelos semánticos explícitos de tipo declarativo (ontologías). En el contexto de los lenguajes de representación del conocimiento, las ontologías representan esquemas que definen formalmente la estructura y las restricciones de la información. Por otra parte, los modelos de información tradicionales contemplan la interacción y manipulación de los objetos de gestión a través de operaciones, acciones e invocación de métodos. Estas formas son las mismas tanto si se actúa directamente sobre los

objetos gestionados, como si se actúa a través de intermediarios (como en el caso de proxies de gestión o pasarelas entre arquitecturas de gestión). Si bien comienza a contemplarse su importancia [Dam02], apenas existen trabajos previos en el área de la especificación de protocolos de interacción (principalmente entre intermediarios) como un modelo de interacción adecuado al paradigma de agentes autónomos con mayor poder expresivo que las operaciones sobre objetos de gestión. Por último, ninguno de los modelos de organización propuestos es adecuado al carácter cooperativo de este paradigma.

Capítulo 3

Modelo de Información: Estructura del Conocimiento de Gestión

Índice General

3.1. Alcance de la propuesta 49

3.2. Conceptualización CIM-OWL 51

3.3. "Mapping" del metaesquema CIM al metamodelo OWL 55

3.4. Adecuación del "Mapping" CIM-OWL a la Lógica Descriptiva 76

3.5. Servicios de razonamiento automático 91

Uno de los principales objetivos de una arquitectura de gestión es ofrecer un marco conceptual para la estandarización de los elementos fundamentales en que se apoyarán posteriormente las plataformas y las soluciones de gestión que se desarrollen en base a la misma. Este marco resulta crucial a la hora de implementar una solución de gestión integrada en un entorno abierto y fuertemente distribuido como es Internet. En el corazón de toda arquitectura de gestión se encuentra por tanto la especificación de la *información de gestión*. Ésta se hace disponible en forma de definiciones de *objetos gestionados* elaboradas siguiendo un modelo determinado. Los objetos gestionados representan una abstracción de aquellas características de los recursos y servicios implicados que se consideran relevantes para el proceso de gestión. Entre los elementos fundamentales que deben ser estandarizados destacan pues los métodos utilizados para el modelado y la descripción de los objetos del dominio de gestión, y que constituyen el *Modelo de Información* de la arquitectura de gestión. El modelo de información debe ofrecer un consenso en el modo en que estos objetos son identificados, cómo se representa su estado, cómo se define su comportamiento y cómo pueden ser manipulados.

El éxito en el proceso de incorporación de agentes autónomos racionales, capaces de razonar e integrar de forma dinámica conocimiento y servicios, depende en gran medida de la evolución de los modelos de información de las arquitecturas de gestión a modelos semánticos explícitos de tipo declarativo u *ontologías* [Góm97] dotados de una sólida base formal. En el contexto de los lenguajes de representación del conocimiento, las ontologías representan esquemas que definen la estructura y las restricciones de la información [Gru93][Gua98].

Por otra parte, los modelos de información tradicionales contemplan la interacción y manipulación de los objetos de gestión a través de operaciones, acciones e invocación de métodos. Estas formas son las mismas tanto si se actúa directamente sobre los objetos gestionados, como si se actúa a través de intermediarios (como en el caso de proxies de gestión o pasarelas entre arquitecturas de gestión). Si bien comienza a contemplarse su importancia [Dam02], apenas existen trabajos previos en el área de la especificación de *protocolos de interacción* (principalmente entre intermediarios) como un modelo de interacción con mayor poder expresivo que las operaciones sobre objetos de gestión.

Por último, si bien la *gestión basada en políticas* está atrayendo cada vez más atención en los últimos años [Dam02], no es sino ahora que está empezando a ser incluida en los modelos de información de las principales arquitecturas de gestión existentes [MES00, Raf02].

En este capítulo se describe el modelo de información que ha sido desarrollado para la arquitectura *Nesmarq* desde la perspectiva de la estructura de información de gestión. Se dejan para el siguiente capítulo los aspectos del mismo relacionados con el modelado de la dinámica de las interacciones entre agentes. El modelo que se presenta (denominado CIMOnt) constituye un modelo conceptual de representación de información formal de tipo semántico. CIMOnt está basado en lógica descriptiva, por lo que permite un mayor grado de automatización en la gestión que los modelos de las arquitecturas tradicionales en base a su utilización por parte de agentes autónomos racionales, capaces de razonar, inferir e integrar de forma dinámica conocimiento y servicios conceptualizados mediante el modelo CIM y formalizados a nivel semántico mediante lógica descriptiva. El modelo desarrollado también permite desarrollar herramientas de modelado visual capaces de razonar acerca de los modelos en tiempo de diseño, a partir de motores de razonamiento automático basados en lógica descriptiva. El modelo de información incluye además un "mapping" a nivel de meta-modelo de CIM al lenguaje de especificación de ontologías OWL, que supone un significativo avance en el área de la representación y el intercambio basado en XML de modelos y meta-información.

3.1. Alcance de la propuesta

En ningún caso se plantea la sustitución de los modelos de información existentes por uno nuevo desarrollado desde cero; la arquitectura de gestión holónica propuesta pretende en su lugar complementar estos modelos con otro nuevo dotado de mayor expresividad tal y como muestra la figura 3.1. El propósito es adaptar estos modelos a las formas de razonamiento y cooperación asociadas al paradigma de agentes autónomos.

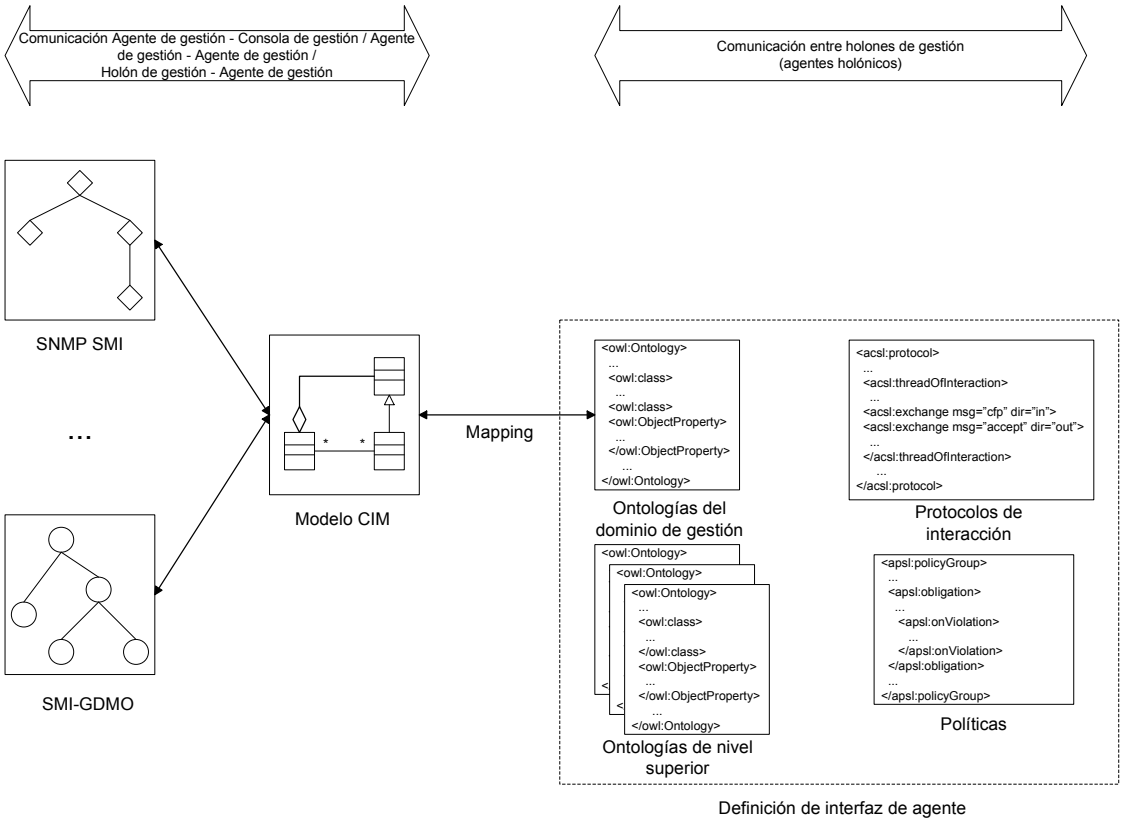

Figura 3.1: Modelo de Información propuesto

La figura 3.1 muestra la incorporación al modelo de información de las arquitecturas existentes de un nuevo elemento denominado *Definición de Interfaz de Agente* (DIA), consecuencia de la evolución del modelo organizativo a un modelo *holónico* como el que se describe más adelante en el capítulo 5, esto es, un modelo heterárquico de tipo cooperativo fundamentado en el concepto de *holón de gestión*. En el contexto de una organización holónica u *holarquía*, los agentes holónicos u holones de gestión interactúan para intercambiar conocimiento y para cooperar y coordinar sus actividades de gestión. La aproximación seguida en esta tesis consiste en describir estas interacciones como *conversaciones* especificadas mediante *protocolos de*

interacción. El conjunto de conversaciones en que puede participar un holón define su interfaz de comunicación (y por tanto de servicio). Por tanto, los conjuntos estandarizados de protocolos de interacción que especifican dichas conversaciones se proponen como *Definiciones de Interfaz de Agentes* (DIAs), del mismo modo que los conjuntos de definiciones de procedimientos y funciones suponen las interfaces y/o bibliotecas de programación (APIs) en otros paradigmas de programación utilizados tradicionalmente en gestión (principalmente los paradigmas de objetos y componentes). En este contexto del modelo de información, en el capítulo 4 se desarrollará la sintaxis abstracta y la semántica operacional de un lenguaje de especificación formal de conversaciones (ACSL) que permitirá describir con claridad y precisión estas interfaces de modo que puedan ser consumidas, tanto por los diseñadores y programadores (generalmente mediante herramientas CASE), como por los propios agentes, de forma automática, durante su interacción.

Además de la especificación del conjunto de conversaciones en que puede participar un holón de gestión, una DIA especifica también el conjunto de *ontologías* que conceptualizan y definen formalmente la estructura, el significado y las restricciones de la información y el conocimiento manejado por dicho holón, de modo que puedan ser transferidas a otros holones y comprendidas por éstos a través de procesos automáticos de razonamiento fundamentados en lógicas descriptivas (LDs). Los modelos de información existentes hasta la fecha están basados fundamentalmente en la orientación a objetos. Si bien XMI y la representación XML de CIM, tal y como se ha comentado anteriormente, constituyen un paso adelante en la especificación transferible e interpretable de forma automática de éstos modelos, resultan inadecuados para el modelado de conocimiento [semántico], el intercambio de estos modelos entre agentes autónomos, su interpretación directa y el razonamiento acerca de los mismos por parte de los agentes. La utilización de ontologías basadas en lógica descriptiva y lenguajes de ontologías como OWL facilitará esta tarea enormemente. Sin embargo, se requiere de la especificación un "mapping" entre ambas representaciones y, más concretamente, ambos lenguajes de modelado. Este es un paso previo en la evolución de los modelos de información orientados a la información hacia modelos de información orientados al conocimiento, más adecuados en arquitecturas de gestión basadas en un paradigma cooperativo. El siguiente paso es la utilización de ontologías expresadas directamente en OWL que aprovechen todo su poder expresivo. Las siguientes secciones describen la faceta estructural del modelo de información.

Por último, una DIA especifica el conjunto de políticas asociadas al rol desempeñado por dicho holón dentro de la organización u holarquía en que participa. Estas

políticas se expresan como parte del modelo de información (y por tanto mediante una ontología), si bien su semántica se define en términos un modelo normativo que forma parte del Modelo de Organización que se describe en el capítulo 5.

3.2. Conceptualización CIM-OWL

Como se deduce de las conclusiones extraídas del análisis del espacio de soluciones aportado en el capítulo 2, el Modelo de Información Común (CIM) desarrollado por el DMTF se muestra como el modelo más adecuado para constituir la base de un modelo más evolucionado orientado al conocimiento. CIM proporciona una aproximación rigurosa al modelado de redes y sistemas distribuidos basada en el paradigma de objetos. El meta-esquema CIM guía la construcción del resto de esquemas CIM y constituye la base para la construcción de un vocabulario adecuado para la descripción y el análisis de los sistemas gestionados.

Por su parte, el lenguaje OWL [1] (Ontology Web Language) constituye un estándar emergente para especificación de ontologías con soporte para procesos de razonamiento trazable e inferencia lógica basados en los creados para las lógicas descriptivas. Su fundamentación en estándares Web (XML, RDF, URIs, etc), no sólo no le restringe al contexto de la Web Semántica, sino que le promete como un candidato idóneo para la especificación del modelo de información de las arquitecturas de gestión basadas en la Web. Tanto DAML+OIL, como su sucesor OWL pueden considerarse como una evolución del RDF-Schema hacia lenguajes de ontologías capaces de proporcionar descripciones semánticas más ricas de los recursos.

El Modelo de Información desarrollado se propone como un "mapping" CIM-OWL e ilustra el papel fundamental de OWL como soporte para el intercambio de información basada en XML expresable de forma natural mediante modelos entidad-

[1] DARPA-KSF crea en 2000 DAML (DARPA Agent Markup Language), un lenguaje de ontologías surgido como extensión al RDF (Resource Description Framework). Este último constituye una aproximación a la representación de conocimiento próxima a las redes semánticas y a los grafos de conceptos. Conceptualmente, RDF representa el conocimiento mediante un grafo cuyos nodos y arcos modelan recursos, propiedades de los recursos y valores de éstas propiedades, si bien admite otras representaciones como ternas sujeto(recurso)-predicado(propiedad)-objeto(valor) y una sintaxis XML. El marco de trabajo RDF incluye también un lenguaje de esquema (RDF-Schema) que permite definir vocabularios RDF y proporcionar así un sistema de tipos (categorías) para RDF. RDF fue creado para definir ontologías ligeras sobre recursos web, por lo que su extensión a DAML obtuvo gran difusión en el contexto del proyecto de Web Semántica, promovido entre otros por el propio DARPA y, posteriormente, por el W3C (World Wide Web Consortium). A su vez, DAML se completó posteriormente con un nivel de inferencia semántica denominado OWL, surgiendo así el lenguaje DAML+OIL. En el último trimestre de 2002 el W3C decide actuar como aglutinador del trabajo llevado a cabo hasta ese momento y propone OWL (Ontological Web Language) como lenguaje de ontologías estándar para la Web Semántica. OWL está fuertemente influenciado por DAML+OIL y deja obsoleto a éste último.

relación o modelos orientados a objetos (clases, atributos y relaciones), así como su adecuación a los procesos de razonamiento propios del paradigma de agentes autónomos. Ilustra también una marcada tendencia hacia el intercambio no sólo de información de gestión sino también de los propios modelos.

Esta sección recoge las características del "mapping" CIM-OWL desarrollado como parte de esta tesis y que se describe en la sección 3.3. En ella se justifica la elección de un "mapping" semántico a nivel de meta-modelo y la relacionan con el resto de posibilidades descartadas.

3.2.1. "Mapping" sintáctico Vs "mapping" semántico

La meta de la gestión integrada es ofrecer una visión uniforme de los sistemas administrados y diseñar una implementación eficiente de las aplicaciones que permita un acceso sin restricciones a estos objetos gestionados a través de las transiciones entre diferentes arquitecturas. La traducción entre modelos de información es necesaria tanto en el caso de que la transición entre arquitecturas se produzca en pasarelas de gestión, como en el caso de que se produzca en la propia plataforma de gestión. Las definiciones de información de gestión elaboradas siguiendo un determinado modelo de información pueden traducirse a otro modelo de información tanto a nivel semántico, como a nivel sintáctico.

La *traducción a nivel sintáctico* busca la elaboración de una función de traducción para los modelos de información de las diferentes arquitecturas y sus correspondientes sintaxis de descripción, de modo que se permita la traducción automática, e incluso estándar, de los elementos de un lenguaje de especificación a los elementos equivalentes en el otro lenguaje de especificación. No se requiere ninguna interpretación de la semántica de la información traducida, por lo que la información de gestión resultante se integra en la arquitectura destino sin que se considere la información de gestión ya existente en esta arquitectura y que tiene una semántica equivalente. La mayoría de traducciones entre modelos de información se han realizado a este nivel, e.g. SMI-GDMO del ISO/CCITT e Internet Management Coexistence (IIMC) [NMF30][NMF26], GDMO-CIM, SMI/GDMO-CORBA IDL/CCM del OMG-TMF Joint Inter-Domain Management (JIDM) que requiere la traducción de tipos de datos ASN.1 utilizados tanto en la gestión OSI como en la gestión Internet a tipos de datos IDL y la traducción de definiciones GDMO y SMI a interfaces IDL [SOH97][OG509].

La *traducción a nivel semántico*, por su parte, persigue el análisis de la semántica presente en las definiciones de información de una arquitectura en un intento por proyectar esta semántica lo más fielmente posible en la información de gestión

existente en la otra arquitectura. El objetivo es conseguir un nivel máximo de integración de la información de gestión de ambas arquitecturas. Las clases abstractas y genéricas de la arquitectura objetivo pueden utilizarse como base del proceso. La traducción de la semántica de una definición requiere el "mapping" de objetos gestionados concretos y sus interrelaciones y proporciona un mayor nivel de integración de las arquitecturas, mientras que la traducción de su sintaxis tan solo requiere el "mapping" de los lenguajes de especificación.

Si bien la integración de la información de gestión de la arquitectura origen en la arquitectura destino es mucho más deseable que la coexistencia en esta última de información semánticamente equivalente (la propia y la proveniente de la arquitectura origen), existe un precio a pagar por ello. Hasta ahora, las traducciones a nivel semántico se han venido realizado "artesanalmente" debido a la dificultad asociada a la traducción automática, que requiere de un análisis profundo de la semántica de la información de ambas arquitecturas, el cual podría ser no determinista. La utilización de ontologías introduce nuevas oportunidades en el proceso de automatización de la traducción a nivel semántico. Se pretende alcanzar un mayor grado de integración que permita no solo acceder fácilmente a información de gestión de otras arquitecturas (que, en todo caso, sigue viéndose como información de otras arquitecturas), sino integrar dicha información como parte de la arquitectura propia.

El modelo de información CIM del DMTF supone un primer intento por integrar a nivel semántico los modelos de información (y las MIBs) de SNMP y de OSI, para lo cual se propone el uso de XML. Sin embargo, el método propueto adolece de utilizar ontologías y lenguajes de ontologías en el proceso de integración, lo cual dificulta la automatización de dicha integración. El "mapping" CIM-OWL propuesto facilitará esta labor al permitir el razonamiento automático acerca de los modelos.

3.2.2. Niveles de "mapeado"

El "mapping" entre el modelo CIM y OWL puede llevarse a cabo a tres niveles diferentes en función de la expresividad deseada para el mismo:

Nivel de meta meta-modelo Este tipo de "mapping" utiliza las construcciones del meta-modelo de la arquitectura destino para describir las construcciones del meta-modelo del modelo de información de la arquitectura origen. En esencia, el meta-modelo de la arquitectura destino se utiliza como meta meta-modelo del modelo de información de dicha arquitectura. La arquitectura SNMP utiliza SMI (Structure of Management Information) como meta-modelo orientado a datos para describir la información de gestión en Internet. La arquitectura OSI-MA de ISO/CCITT utiliza GDMO (Guidelines for the

SMI	**GDMO**
Grupos	Clases de objetos gestionados
Definiciones tabulares	Sin traducción
Filas (tablas)	Clases de objetos
Resto de objetos	Atributos de la clase correspondiente
Alarmas (traps)	Notificación genérica enviada por la clase que representa el grupo *system*
CMIS Service feature synchronization	Semántica atómica y best-effort.

Cuadro 3.1: Reglas de traducción SMI-GDMO

Definition of Managed Objects) como meta-modelo orientado a objetos, del mismo modo que la DMTF utiliza MIF (Management Information Format), la iniciativa WBEM utiliza CIM, TINA utiliza ODL (Object Description Language) o el OMG utiliza para su OMA CCM+IDL. Tiene sentido por tanto, describir un conjunto de "mappings" de técnica en los que el meta-modelo de cada una de estas arquitecturas actúe como meta meta-modelo capaz de describir las sintaxis de las restantes (SMI, GDMO, CIM-MOF y CIM/XML y MIF). Otra opción es aprovechar el papel de CIM como lenguaje intermedio con el que efectuar traducciones a nivel semántico, de modo que se facilite la traducción entre las restantes arquitecturas, incluso las emergentes.

Nivel de meta-modelo Este "mapping" proyecta las construcciones de la arquitectura origen en meta construcciones de la arquitectura destino, de modo que cualquier modelo expresado según el modelo de información de la arquitectura origen pueda traducirse al modelo de información de la arquitectura destino. El mayor esfuerzo de diseño requerido por este "mapping" se centra en el desarrollo de un "mapping" entre el meta-modelo origen y el meta-modelo destino.

Nivel de modelo Este "mapping" toma un modelo expresado según el modelo de información de la arquitectura origen y mapea su contenido a un modelo en la arquitectura destino. No se presupone la existencia de un "mapping" a nivel de meta-modelo; se trata principalmente de un "mapping" de contenido. Se puede ver como la reexpresión del contenido de un modelo utilizando una técnica con mayor poder expresivo.

Los cuadros 3.3, 3.1 y 3.2 muestran los principales elementos de los "mappings" existentes hasta donde alcanza el conocimiento del autor.

GDMO	IDL
Clases de objetos gestionados	Interfaces IDL (con propagación de las relaciones de herencia
Atributos	Operaciones de acceso (de acuerdo con los tipos de acceso permitidos
Operaciones (ACTION)	Operaciones IDL
Notificaciones	Interfaces IDL con soporte para la notificación de sucesos del servicio COS-EventService de CORBA y los modelos de envío push/pull

Cuadro 3.2: Reglas de traducción GDMO-IDL

MIF	CIM
Atributos DMI	Propiedades CIM
Atributos clave DMI	Propiedades clave CIM
Grupos DMI	Clases CIM
Componentes DMI	Clases CIM

Cuadro 3.3: Reglas de traducción MIF-CIM

3.3. "Mapping" del metaesquema CIM al metamodelo OWL

En las siguientes subsecciones se describe de manera sistemática el "mapping" a nivel de meta-modelo que ha sido desarrollado con el fin de poder proyectar las meta-construcciones del lenguaje de modelado utilizado por el DMTF para el desarrollo del modelo de información CIM, y que constituyen el, así denominado, metaesquema CIM en terminología DMTF, en meta-construcciones del meta-modelo del lenguaje OWL. Este "mapping" permite expresar cualquier modelo de información CIM como un modelo de información OWL que preserva su semántica, esto es, una ontología OWL que permite su transporte e interpretación automática por parte de los holones de gestión. El objetivo es permitir a los holones de gestión alcanzar acuerdos explícitos de carácter semántico acerca de los modelos de información que utilizan.

Generalmente, cuando se "mapea" el contenido de los modelos expresados según el modelo de información de una arquitectura entre diferentes meta-modelos, se pierde información. Los meta atributos introducidos en el vocabulario OWL permiten extender el meta modelo OWL y resultan especialmente útiles a la hora de eliminar estas pérdidas.

Se dice que dos modelos M_1 y M_2 son semánticamente equivalentes (y se expresa $M_1 \equiv M_2$) si existe una correspondencia biunívoca entre las instancias de M_1 y las instancias de M_2 que preserva las relaciones entre dichas instancias, de modo que los modelos tan solo difieran en aspectos intrascendentes tales como el nombrado,

la ordenación no significativa de construcciones o la inclusión de redundancia.

Así, el "mapping" desarrollado del lenguaje de modelado utilizado por CIM L_{CIM} al lenguaje de modelado OWL L_{OWL} es un "mapping" de equivalencia semántica, ya que constituye una función *map* de los modelos de L_{CIM} en los modelos de L_{OWL} que preserva la equivalencia semántica entre dichos modelos. Esto es,

$$\forall m_1 \in L_{CIM}, \forall m_2 \in L_{OWL} \cdot m_1 \equiv m_2 \rightarrow map(m_1) \equiv map(m_2)$$

Con el fin de ilustrar la especificación se describe de forma incremental el proceso de traducción del modelo de políticas (CIM Core Policy Model) mostrado en la figura 3.2. El ejemplo muestra también cómo los meta-atributos introducidos en el vocabulario OWL permiten extender el meta modelo OWL y resultan especialmente útiles a la hora de minimizar las pérdidas de semántica ocasionadas en el proceso de traducción. El apéndice A recoge la traducción completa.

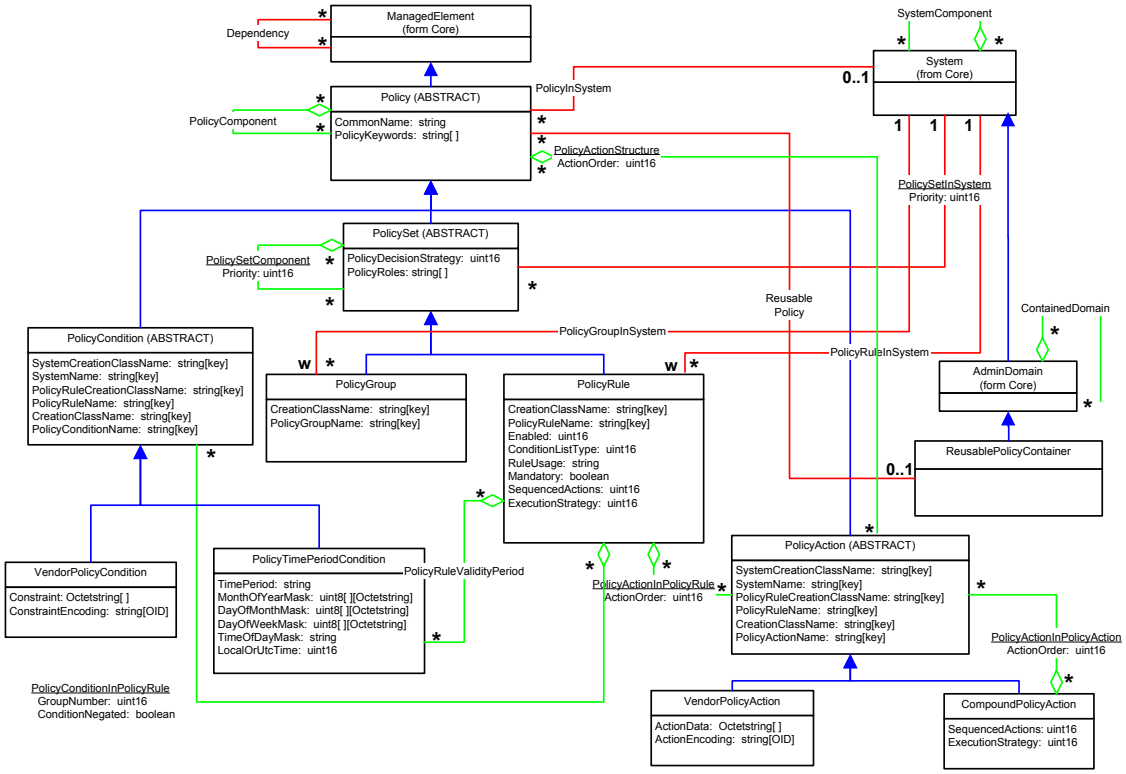

Figura 3.2: Modelo de políticas CIM versión 2.6 (DMTF Service Level Agreements WG)

3.3.1. Extensiones al OWL-DL requeridas para el "mapping" CIM-OWL

La especificación de OWL define dos subconjuntos específicos del lenguaje:

OWL Lite diseñado con el fin de potenciar el uso de OWL y facilitar la creación de herramientas con funcionalidad incremental gracias a las restricciones sintácticas que impone, y

OWL DL específico para la creación de sistemas de razonamiento automático basados en lógicas descriptivas.

Sin embargo, las características del "mapping" han requerido la extensión de *OWL DL* con algunas características satisfechas tan solo por la especificación global de OWL, también denominada *OWL Full* pese a que ésta viola determinadas restricciones de los razonadores basados en lógica descriptiva, por lo que es poco probable que un sistema de razonamiento automático soporte razonamiento de tipo completo para todas sus características. A continuación se exponen de manera justificada las extensiones a *OWL DL* requeridas por el "mapping" CIM-OWL propuesto y se demuestra la adecuación de la semántica *OWL Full* para satisfacer estas necesidades:

3.3.1.1. Propiedades *DatatypeProperty* con semántica de clave

Las propiedades *DatatypeProperty* y *ObjectProperty* constituyen conjuntos disjuntos en *OWL DL*, de lo que se infiere que sólo las propiedades *ObjectProperty* podrán definirse como propiedades inversamente funcionales (*InverseFunctionalProperty*), simétricas (*SymmetricProperty*) o transitivas (*TransitiveProperty*). Sin embargo, el "mapping" CIM-OWL propuesto requiere la utilización de *InverseFunctionalProperty* para denotar la semántica asociada a las propiedades CIM que constituyen la clave de una clase CIM: propiedades OWL cuyo inverso es funcional e inyectivo. Puesto que estas propiedades deben ser mapeadas en general a *DatatypeProperty*, la restricción impuesta por *OWL DL* resulta un impedimento insalvable. *OWL Full* permite relajar esta restricción de modo que las propiedades *DatatypeProperty* pueden definirse como propiedades inversamente funcionales.

Demostración

El esquema RDF para OWL no restringe en ningún momento *DatatypeProperty* como subclase de *ObjectProperty*. Sin embargo, dada una interpretación *OWL* $I = \langle R_I, P_I, EXT_I, S_I, L_I, LV_I \rangle$ de un vocabulario $V = V_{RDF} \cup V_{RDFS} \cup V_{OWL}$, en la que R_I es el universo de discurso, $P_I \subset R_I$ son las propiedades de I, EXT_I es un "mapping" de P_I en $\wp(R_I \times R_I)$, S_I es un "mapping" de V en R_I (referencias URI en

sus denotaciones), L_I es un "mapping" de literales tipados en LV_I (LV_I contiene al menos todos los valores para literales planos), y sean $CEXT_I = \{x \in R_I \mid \langle x, c\rangle \in EXT_I(S_I(rdf : type))\}$ y $C_I = CEXT_I(S_I(rdfs : Class))$, se satisfacen (entre otras) las siguientes restricciones [WG03a]:

$$c \in CEXT_I(S_I(owl : InverseFunctionalProperty)) \leftrightarrow \\ c \in IOOP \wedge \langle x, y_1\rangle, \langle x, y_1\rangle \in EXT_I(c) \rightarrow x_1 = x_2$$

y

$$CEXT_I(S_I(owl : DatatypeProperty)) = IODP \wedge IODP \subseteq P_I$$

y, por definición, una interpretación *OWL Full* de un vocabulario V satisface además $IOOP = P_I$, por lo que puede deducirse que

$$IOOP = P_I \wedge IODP \subseteq P_I \rightarrow IODP \subseteq IOOP$$

y, por tanto,

$$c \in IODP \wedge \langle x, y_1\rangle, \langle x, y_1\rangle \in EXT_I(c) \rightarrow x_1 = x_2 \rightarrow \\ c \in CEXT_I(S_I(owl : InverseFunctionalProperty))$$

lo cual implica que, en *OWL Full* las propiedades *DatatypeProperty* son un subconjunto de las propiedades *ObjectProperty* (trasladable en cierto sentido al concepto de subclase) y por tanto, pueden definirse como *InverseFunctionalProperty*.

Nótese que, en todo caso, las propiedades *DatatypeProperty* seguirán obviamente sin poder definirse como propiedades simétricas (*SymmetricProperty*) ni como propiedades transitivas (*TransitiveProperty*) ya que en ambos casos se requiere que coincidan dominio y rango, lo cual es obviamente imposible en propiedades *DatatypeProperty*.

3.3.1.2. Propiedades acerca de otras propiedades

Todo el vocabulario OWL se define en el así denominado "universo OWL", que constituye una división de un subconjunto del universo RDF en tres partes: individuos (extensión de la clase *Thing*), clases (extensión de la clase *Class*) y propiedades (unión de extensiones de las clases *ObjectProperty*, *DatatypeProperty*). En *OWL Full* las tres partes se identifican con sus homólogas RDF (*rdfs:Resource*, *rdfs:Class* y *rdf:Property*), por lo que sus extensiones no constituyen conjuntos disjuntos. El "mapping" CIM-OWL propuesto precisa de esta propiedad de *OWL Full* para poder definir "metapropiedades", esto es, propiedades OWL cuyo dominio es a su vez una propiedad OWL, con el fin de poder "mapear" las asociaciones binarias con propiedades CIM sin necesidad de utilizar una clase OWL intermedia. La sección 3.3.8

detalla la utilización de metapropiedades OWL en el "mapping" y precisa aún más esta motivación.

Demostración

Dada la interpretación *OWL I* expresada sobre estas líneas, se satisfacen (entre otras) las siguientes restricciones:

$$e \in CEXT_I(S_I(owl : ObjectProperty)) \rightarrow EXT_I(e) \subseteq IOT \times IOT$$

y

$$e \in CEXT_I(S_I(owl : DatatypeProperty)) \rightarrow EXT_I(e) \subseteq IOT \times LV_I$$

y, por definición, una interpretación *OWL Full* de un vocabulario V satisface además $IOT = R_I$, por lo que, puesto que $P_I \subset R_I$, se deduce $P_I \subset R_I = IOT$, con lo que una propiedad OWL puede ser dominio de otra propiedad OWL en *OWL Full*. (En *OWL DL* sólo puede ser $IOT \subseteq R_I$).

3.3.1.3. Conjuntos de propiedades con semántica de clave

Una clase CIM puede contener un conjunto de propiedades calificadas como clave ([KEY]). Estas propiedades constituyen su clave, generalmente propagada desde otra clase a través de una asociación CIM. En una primera aproximación, podría pensarse en "mapear" directamente cada una de estas propiedades CIM a una sentencia *DatatypeProperty* de tipo *InverseFunctionalProperty*. Sin embargo, esta aproximación no recoge la semántica asociada al concepto de clave CIM, ya que ninguna de las claves propagadas constituye por si misma una clave, y sí el conjunto de todas ellas. Por tanto, el "mapping" propuesto utiliza una lista (rdf:Bag en función de si importa o no el orden a la hora de construir la clave) de propiedades *DatatypeProperty* (una por cada propiedad CIM implicada) para modelar la clave de una clase CIM y una propiedad *DatatypeProperty* de tipo *InverseFunctionalProperty* denominada *ccm:Key* que liga para expresar que el conjunto de dichas propiedades constituyen la clave de la clase.

A continuación se muestra, a modo de ejemplo, un fragmento del *Modelo de Políticas CIM* recogido en el apéndice A, correspondiente a la definición de la clave compuesta *PolicyRuleKey* formada por las propiedades *CreationClassName* y *PolicyRuleName*.

```
<owl:DatatypeProperty rdf:ID="CreationClassName">
   <rdfs:comment>DatatypeProperty para la propiedad CreationClassName.
                 Clave</rdfs:comment>
   <rdfs:range rdf:resource="xsd:string"/>
</owl:DatatypeProperty>
```

```
<owl:DatatypeProperty rdf:ID="PolicyRuleName">
   <rdfs:comment>DatatypeProperty para la propiedad PolicyRuleName.
                 Clave</rdfs:comment>
   <rdf:type rdf:resource="owl:InverseFunctionalProperty"/>
   <rdfs:range rdf:resource="xsd:string"/>
</owl:DatatypeProperty>

<rdf:Bag ID="PolicyRuleKeyBag">
   <rdf:li resource="#CreationClassName"/>
   <rdf:li resource="#PolicyRuleName"/>
</rdf:Bag>

<ccm:KeyProperty rdf:ID="PolicyRuleKey">
   <rdfs:domain rdf:resource="ccm:System"/>
   <rdfs:range rdf:resource="#PolicyRuleKeyBag"/>
</ccm:KeyProperty>
```

Demostración

Dada la interpretación *OWL I* expresada sobre estas líneas, se satisface (entre otras) la siguiente condición de comprensión sobre listas:

$$\forall x_i, \ldots, x_j \in IOT \cdot (I = \{x_1, \ldots, x_n\} \rightarrow \exists y \in IOC \cdot \langle y, I \rangle \in EXT_I(S_I(owl : oneOf))$$

y, por definición, una interpretación *OWL Full* de un vocabulario V satisface además $IOT = R_I$, puesto que IOT en *OWL Full* es el universo de discurso RDF completo, por lo que se deduce que *OWL Full* admite, de una parte, la generación de listas de cualquier tipo, incluyendo listas de propiedades ya que $P_I \subset R_I = IOT$ y, de otra parte, la utilización de estas listas como rango de una propiedad *ObjectProperty*.

3.3.1.4. Herencia simple

Una clase CIM puede tener a lo sumo una superclase, por lo que ha sido necesario extender el esquema RDF de OWL con una nueva propiedad *FunctionalSubClassOf* que restringe la propiedad OWL *subClassOf* de modo que sea funcional:

```
<rdf:Property rdf:ID="functionalSubClassOf">
  <rdfs:label>functionalSubClassOf</rdfs:label>
  <rdf:type rdf:resource="#FunctionalProperty"/>
  <rdfs:subPropertyOf rdf:resource="&rdf;subClassOf"/>
  <rdfs:domain rdf:resource="#Class"/>
  <rdfs:range rdf:resource="#Class"/>
</rdfs:Class>
```

No es necesario ampliar la semántica OWL.

Demostración

Dada la interpretación *OWL I* expresada sobre estas líneas, se satisface la siguiente restricción para propiedades *FunctionalProperty*

CIM	**OWL**
Schema	owl:Ontology
Schema@schemaName	owl:Ontology/@rdfs:about

Cuadro 3.4: "Mapping" de esquemas CIM

$$c \in CEXT_I(S_I(owl : FunctionalProperty)) \leftrightarrow \\ c \in IOOP \cup IODP \wedge \langle x, y_1 \rangle, \langle x, y_2 \rangle \in EXT_I(c) \rightarrow y_1 = y_2$$

y, de la definición de *FunctionalSubClassOf* dada sobre estas líneas se tiene que

$$owl : functionalSubClassOf \in EXT_I(S_I(owl : FunctionalProperty))$$

por lo que

$$c \in CEXT_I(S_I(owl : functionalSubClassOf)) \leftrightarrow \\ c \in IOOP \wedge \langle x, y_1 \rangle, \langle x, y_2 \rangle \in EXT_I(c) \rightarrow y_1 = y_2$$

por lo que la nueva propiedad adquiere la semántica asociada a las propiedades funcionales.

Por otra parte, de la definición de *FunctionalSubClassOf* dada sobre estas líneas se tiene que

$$\forall x, y \in IOC \cdot \langle x, y \rangle \in EXT_I(S_I(owl : functionalSubClassOf)) \rightarrow \\ \langle x, y \rangle \in EXT_I(S_I(owl : subClassOf))$$

por lo que se conserva su semántica original.

OWL Full permite que una ontología aumente el significado del vocabulario (RDF u OWL) predefinido, por lo que se utiliza en la medida en que las necesidades de meta-modelado lo requieren.

3.3.2. "Mapping" de esquemas CIM

Un *esquema* CIM es un conjunto de clases relacionadas, asociadas a un único creador. Los esquemas resultan útiles para conseguir unicidad en el nombrado de clases y para organizar la información de gestión. Un esquema CIM se mapea directamente a una ontología OWL (Ontology). El cuadro 3.4 recoge los elementos fundamentales del "mapping".

A continuación se muestra la descripción OWL del esquema *CIM Core Policy Model Schema* descrito en la figura 3.2. Se ha utilizado el estándar *Dublin Core Metadata* [DUB03].

CIM	**OWL**
CLASS	owl:Class
CLASS@className	owl:Ontology/@rdf:ID

Cuadro 3.5: "Mapping" de clases CIM

```
<owl:Ontology rdf:about='urn:TIC2001-3451:ontologies:cim-policy-model#'>
  <rdfs:comment>Ontologia para el modelo de politicas de CIM</rdfs:comment>
  <owl:versionInfo>
          $Id: CIMPolicyModelOntology v 1.0 02/02/2003 Javier Soriano$
      </owl:versionInfo>
  <owl:imports rdf:resource='http://www.w3.org/2002/07/owl'/>
  <owl:imports rdf:resource='http://www.w3.org/2000/01/rdf-schema'/>
  <dc:title>OWL CIM Policy Model Ontology</dc:title>
  <dc:creator>Francisco Javier Soriano Camino</dc:creator>
  <dc:subject>OWL; Ontologies; Policy Models</dc:subject>
  <dc:description>Ontologia OWL para el modelo de politicas CIM</dc:description>
  <dc:publisher>Grupo de Redes y Sistemas Distribuidos. FIM</dc:publisher>
  <dc:date>March 02, 2003</dc:date>
  <dc:format>text/xml</dc:format>
  <dc:language>es</dc:language>
  <dc:identifier>urn:TIC2001-3451:ontologies:cpm</dc:identifier>
</owl:Ontology>
```

3.3.3. "Mapping" de clases CIM

Una *clase* CIM es una colección de instancias de un mismo tipo, esto es, con las mismas propiedades (y métodos). Una clase CIM se mapea directamente a una clase OWL (Class). El cuadro 3.5 recoge los elementos fundamentales del "mapping".

A continuación se muestra la descripción OWL de la clase *PolicyRule*.

```
<owl:Class rdf:ID='PolicyRule'>
  <rdfs:comment>Clase PolicyRule</rdfs:comment>
  <rdfs:label xml:lang='en'>policy-rule</rdfs:label>
  <rdfs:label xml:lang='es'>regla</rdfs:label>
  <owl:FunctionalSubClassOf rdf:resource='#PolicySet'/>
</owl:Class>
```

3.3.4. "Mapping" de jerarquías de generalización

Las clases CIM se organizan en jerarquías de generalización mediante la relación de herencia (única). Así, una jerarquía de generalización representa las relaciones "es subtipo de" existentes entre las diferentes clases que participan en un modelo CIM, por lo que se mapea en una jerarquía OWL en la que, por cada relación de herencia en la jerarquía CIM, se origina una propiedad de subclase (*subClassOf*) cuyo dominio (*domain*) es la clase hija y cuyo rango (*range*) es la clase padre de dicha relación. El cuadro 3.6 recoge los elementos fundamentales del "mapping".

CIM	**OWL**
Generalización CIM	owl:functionalSubClassOf
: CLASS@className	owl:subClassOf/@rdf:resource

Cuadro 3.6: "Mapping" de jerarquías de generalización CIM

Puesto que una clase CIM puede tener a lo sumo una superclase, ha sido preciso restringir la propiedad *subClassOf* de modo que sea funcional. Con este fin se ha extendido el esquema RDF de OWL con la siguiente sentencia:

```
<rdf:Property rdf:ID="FunctionalSubClassOf">
  <rdfs:label>FunctionalSubClassOf</rdfs:label>
  <rdf:type rdf:resource="#FunctionalProperty"/>
  <rdfs:subPropertyOf rdf:resource="&rdf;subClassOf"/>
  <rdfs:domain rdf:resource="#Class"/>
  <rdfs:range rdf:resource="#Class"/>
</rdfs:Class>
```

En la sección 3.3.1.4 se demostró que esta sentencia aporta la semántica requerida sin necesidad de extender la semántica de OWL.

La descripción OWL de la clase CIM *PolicyRule* mostrada en la sección anterior especificaba la relación de herencia simple entre esta clase y la clase CIM *PolicySet*.

3.3.5. "Mapping" de propiedades CIM

Una *propiedad* de clase CIM es un valor utilizado para caracterizar diferentes instancias de la clase en cuestión. Las propiedades de clase se mapean directamente en propiedades de tipo de datos OWL (*DataTypeProperty*), tal y como se explica en el cuadro 3.7. El nombre del atributo se toma como identificador (*rdf:ID*) y como etiqueta (*rdf:label*) de la propiedad, el nombre de la clase que contiene el atributo se toma como dominio (*domain*) y el tipo de datos del atributo se toma como rango (*range*).

La semántica asociada al hecho de que una misma propiedad CIM pueda propagarse a otra(s) clase(s) requiere que no se defina el dominio de la *DatatypeProperty* de manera global, sino a través de restricciones locales (*Restriction*) sobre cada una de las clases implicadas.

Las propiedades cuyo tipo de datos sea otra clase, se tratan como asociaciones binarias, esto es, se mapean en propiedades de objetos (*ObjectProperty*), tal y como se recoge en el cuadro 3.7.

A continuación se muestra la descripción OWL de las propiedades *CreationClassName*, *Mandatory* y *PolicyRoles*.

```
  <owl:DatatypeProperty rdf:ID='CreationClassName'>
```

CIM	**OWL**
Property	owl:DatatypeProperty | owl:ObjectProperty
Property@propertyName	owl:DatatypeProperty/@rdf:ID y owl:DatatypeProperty/rdf:type@rdf:resource=owl:FunctionalProperty
Property@key	owl:DatatypeProperty/rdf:type@rdf:resource=owl:InverseFunctionalProperty
Property@dataType	owl:DatatypeProperty/rdf:type@rdf:resource
Property@array[]	owl:DatatypeProperty@rdf:minCardinality=1 y owl:Restriction/owl:onProperty@rdf:minCardinality=1
Property@array[n]	owl:DatatypeProperty@rdf:cardinality=n y owl:Restriction/owl:onProperty@rdf:cardinality=n
CLASS/Property	owl:Restriction/rdf:onProperty@rdf:cardinality=1
CLASS/Property@array[]	owl:Restriction/rdf:onProperty@rdf:minCardinality=1

Cuadro 3.7: "Mapping" de propiedades CIM

```
  <rdfs:comment>DatatypeProperty para la propiedad CreationClassName.
                Clave</rdfs:comment>
  <rdf:type rdf:resource='owl:InverseFunctionalProperty'/>
  <rdfs:range rdf:resource='xsd:string'/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID='Mandatory'>
  <rdfs:comment>DatatypeProperty para la propiedad Mandatory</rdfs:comment>
  <rdf:type rdf:resource='owl:FunctionalProperty'/>
  <rdfs:range rdf:resource='xsd:boolean'/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID='PolicyRole'>
  <rdfs:comment>DatatypeProperty para la propiedad PolicyRoles.
  Lista</rdfs:comment>
  <rdfs:range rdf:resource='xsd:string'/>
</owl:DatatypeProperty>
```

La propiedad *CreationClassName* es una cadena de caracteres y forma parte de la clave (key) de las clases que la referencian, lo cual hace que la *DatatypeProperty* asociada tenga por rango el tipo de datos XMLSchema *xsd:string* y por tipo *InverseFunctionalProperty*, respectivamente. La propiedad *mandatory* es de tipo booleano y toma a lo sumo un único valor, por lo que la *DatatypeProperty* asociada es de tipo *FunctionalProperty* y tiene por rango el tipo de datos XMLSchema *xsd:boolean*, tal y como se especifica en el cuadro de equivalencias de tipo CIM-OWL recogida en el apéndice A. Por último, la propiedad *PolicyRoles* es un array de cadenas de caracteres, por lo que se le asocia una *DatatypeProperty* que tan solo especifica *xsd:string* como rango. El hecho de que se trata de un array se especifica por separado a la hora de asociar las propiedades a la clase que posee el atributo "mapeado":

```
<owl:Class rdf:about='PolicyRule'>
  <rdfs:comment>Propiedad CreationClassName, clave</rdfs:comment>
  <owl:Restriction>
```

```
    <owl:onProperty rdf:resource='#CreationClassName'/>
    <owl:cardinality>1</owl:cardinality>
  </owl:Restriction>
</owl:Class>
<owl:Class rdf:about='PolicyRule'>
  <rdfs:comment>Propiedad Mandatory</rdfs:comment>
  <owl:Restriction>
    <owl:onProperty rdf:resource='#Mandatory'/>
    <owl:cardinality>1</owl:cardinality>
  </owl:Restriction>
</owl:Class>
<owl:Class rdf:about='PolicySet'>
  <rdfs:comment>Propiedad PolicyRoles. Lista</rdfs:comment>
  <owl:Restriction>
    <owl:onProperty rdf:resource='#PolicyRole'/>
    <owl:minCardinality>1</owl:minCardinality>
  </owl:Restriction>
</owl:Class>
```

Se observa que las dos primeras propiedades utilizan el axioma *cardinality* para ligarse como tales a la clase *PolicyRule*, mientras que la tercera utiliza el axioma *minCardinality* para ligarse a la clase *PolicySet*, con el propósito de especificar su carácter de array.

Como se ha comentado previamente, el hecho de que el "mapping" propuesto no determine un dominio a la hora de especificar una *DatatypeProperty* permite expresar la semántica de las asociaciones [weak] que vinculan los valores de los atributos clave entre dos clases diferentes. Así, la propiedad CreationClassName aparece también en las clases *PolicyAction* y *PolicyCondition*, además de en *PolicyRule*, tomando en todos los casos valor de la clase *System* a través de las asociaciones *PolicyActionInPolicyRule*, *PolicyConditionInPolicyRule* y *PolicyRuleInSystem* respectivamente, de forma transitiva:

```
<owl:Class rdf:about='PolicyAction'>
  <rdfs:comment>Propiedad CreationClassName, clave</rdfs:comment>
  <owl:Restriction>
    <owl:onProperty rdf:resource='#CreationClassName'/>
    <owl:cardinality>1</owl:cardinality>
  </owl:Restriction>
</owl:Class>
<owl:Class rdf:about='PolicyCondition'>
  <rdfs:comment>Propiedad CreationClassName, clave</rdfs:comment>
  <owl:Restriction>
    <owl:onProperty rdf:resource='#CreationClassName'/>
    <owl:cardinality>1</owl:cardinality>
  </owl:Restriction>
</owl:Class>
```

3.3.6. "Mapping" de asociaciones CIM

Una *asociación* CIM representa una relación semántica entre dos o más objetos mediante una clase que contiene dos o más referencias. Las referencias definen el [nombre del] rol desempeñado por cada objeto en el contexto de la asociación y contienen los atributos de la conexión de la asociación con dicho objeto. Una asociación permite expresar múltiples instancias de una relación para un objeto dado (cardinalidad). Una [clase] asociación sólo puede ser una subclase de otra [clase] asociación. Sólo las asociaciones pueden presentar referencias. La forma en que se definen las asociaciones en CIM (como clases) permite establecer una relación entre clases sin afectar a ninguna de las clases relacionadas. En CIM, las asociaciones no tienen ninguna otra significación, puesto que no afectan a la interfaz de las clases relacionadas.

En el proceso de traducción a OWL de una asociación CIM se distinguen dos situaciones diferentes en función de si se trata de una asociación binaria (con o sin atributos) o una asociación n-aria (n>2).

3.3.6.1. "Mapping" de asociaciones binarias

El "mapping" de una asociación binaria a OWL genera tres propiedades de objeto (objectProperty):

- El objeto asociación se mapea directamente en una propiedad de objeto OWL (*objectProperty)*. El nombre del objeto asociación se utiliza como identificador (*rdf:ID*) de la propiedad y como etiqueta (*rdf:label*), el nombre de la clase OWL fuente constituirá el dominio de dicha propiedad y el nombre de la clase OWL destino constituirá su rango. En CIM, las asociaciones son clases que heredan de *NamedElement*, por lo que siempre tienen nombre.

- De entre los atributos asociados a una referencia, resultan relevantes a la hora de "mapear" dicha referencia a OWL su clase conectada, su nombre y su navegabilidad. Todas las referencias con nombre se mapean a propiedades de objeto (se asume que la existencia de un nombre implica navegabilidad). Como identificador y etiqueta de esta propiedad se utiliza el nombre de la referencia (en el identificador, se antepone el nombre de la asociación, ya que el metamodelo CIM no asegura unicidad de nombre en las referencias). Como rango se utilizará el nombre de la clase ligada al rol y como dominio se utilizará el nombre de la clase ligada al rol opuesto. En CIM, todas las referencias heredan de *NamedElement*, por lo que tienen nombre. La propiedad creada a partir de

CIM	**OWL**
ASSOCIATION	owl:ObjectProperty
ASSOC@name	owl:ObjectProperty/@rdf:ID
ASSOC/Reference@name	owl:ObjectProperty/@rdf:ID y (owl:ObjectProperty@owl:samePropertyAs u owl:ObjectProperty@owl:inverseOf)
ASSOC[/Reference]@multilicity=0..1	owl:FunctionalProperty
ASSOC[/Reference]@multilicity=1	owl:FunctionalProperty owl:Restriction/owl:OnProperty@cardinality=1
ASSOC[/Reference]@multilicity= m..m	owl:ObjectProperty owl:Restriction/owl:OnProperty@cardinality=m
ASSOC[/Reference]@multilicity= n..m	owl:ObjectProperty y owl:Restriction/owl:OnProperty@minCardinality=n y owl:Restriction/owl:OnProperty@maxCardinality=m
ASSOC[/Reference]@W	owl:FunctionalProperty/rdf:type@rdf:resource=WeakProperty

Cuadro 3.8: "Mapping" de asociaciones CIM

la referencia que presenta la misma navegabilidad que la dirección de la asociación se asume como subpropiedad de la propiedad creada en función de la asociación, por lo que se le atribuye una propiedad *subpropertyOf*. Del mismo modo, la propiedad creada a partir de la referencia que presenta una navegabilidad opuesta a la dirección de la asociación se asume como propiedad inversa de las otras dos, por lo que se le atribuyen sendas propiedades *inverseOf*.

A continuación se muestra la descripción OWL de las asociaciones *PolicyRuleInSystem* y *PolicyActionInPolicyRule* desde la perspectiva de la referencia ligada a *PolicyRule*:

```
<owl:FunctionalProperty rdf:ID='PolicyRuleInSystem'>
  <rdfs:comment>ObjectProperty para la asociacion
                PolicyRuleInSystem</rdfs:comment>
  <rdfs:domain rdf:resource='#PolicyRule'/>
  <rdfs:range rdf:resource='ccm:System'/>
</owl:FunctionalProperty>
 <owl:ObjectProperty rdf:ID="PolicyRuleAggregatesPolicyAction">
    <rdf:type rdf:resource="#AggregationProperty"/>
    <rdfs:domain rdf:resource="#PolicyRule"/>
    <rdfs:range rdf:resource="#PolicyAction"/>
    <owl:inverseOf rdf:resource="#PolicyActionInPolicyRule"/>
 </owl:ObjectProperty>
```

Puesto que la cardinalidad de la asociación *PolicyRuleInSystem* ligada a la referencia *System* es 1..1, su *ObjectProperty* asociada es de tipo *FunctionalProperty*.

Este hecho se especifica declarando la propiedad directamente como una propiedad *FunctionalProperty* ya que esta es una subclase OWL de *ObjectProperty*. Obsérvese que esto no ocurría con *DatatypeProperty*, por lo que se utilizaba *rdf:type*. En el caso de *PolicyActionInPolicyRule* se observa que la direccionalidad de la propiedad hace que desde la perspectiva de la referencia ligada a *PolicyRule* deba declararse una propiedad OWL inversa a la propiedad que llevará el mismo nombre que la asociación CIM. También ha sido preciso especificar que se trata de una agregación.

Obsérvese que el "mapping" a OWL permite establecer una asociación CIM entre clases pertenecientes a diferentes esquemas. Es el caso de la clase System, que se define en el esquema *CIM Core Model*, y por tanto en la ontología

```
<owl:Ontology
rdf:about='urn:TIC2001-3451:ontologies:cim-core-model#'>
```

y aun así puede referenciarse en la asociación *PolicyRuleInSystem* haciendo uso del espacio de nombres XML *ccm*:

```
<rdf:RDF xmlns:ccm='urn:TIC2001-3451:ontologies:cim-core-model#'>
```

Una vez definidas las propiedades, es necesario ligarlas a sus respectivos dominios:

```
 <owl:Class rdf:about='PolicyRule'>
   <rdfs:comment>Asociacion PolicyRuleInSystem. 1..1</rdfs:comment>
   <owl:Restriction>
     <owl:onProperty rdf:resource='#PolicyRuleInSystem'/>
     <owl:allValuesFrom rdf:resource='ccm:System'/>
   </owl:Restriction>
   <owl:Restriction>
     <owl:onProperty rdf:resource='#PolicyRuleInSystem'/>
     <owl:cardinality>1</owl:cardinality>
   </owl:Restriction>
 </owl:Class>
  <owl:Class rdf:about='PolicyRule'>
     <rdfs:comment>Asociacion PolicyRuleInPolicyAction. 0..*</rdfs:comment>
     <owl:Restriction>
        <owl:onProperty rdf:resource='#PolicyRuleAggregatesPolicyAction'/>
        <owl:allValuesFrom rdf:resource='#PolicyAction'/>
     </owl:Restriction>
  </owl:Class>
```

Nótese que el hecho de definir *PolicyRuleInSystem* como *FunctionalProperty* le confiere una cardinalidad 0..1, por lo que ha sido necesario restringir ésta a 1..1 haciendo uso del axioma *cardinality*. Del mismo modo, al no especificar restricciones de cardinalidad sobre *PolicyRuleInPolicyGroup* se refleja el hecho de que su cardinalidad es 0..*.

Resta pues especificar estas mismas asociaciones desde la perspectiva de las referencias *System* y *PolicyAction*, respectivamente, ya que en OWL las *ObjectProperty* siempre son dirigidas:

```
<owl:Class rdf:about='PolicyAction'>
   <rdfs:comment>Asociacion PolicyActionInPolicyRule. 0..*</rdfs:comment>
   <owl:Restriction>
      <owl:onProperty rdf:resource='#PolicyActionInPolicyRule'/>
      <owl:allValuesFrom rdf:resource='#PolicyRule'/>
   </owl:Restriction>
</owl:Class>
<owl:ObjectProperty rdf:ID='PolicyActionInPolicyRule'>
   <rdfs:domain rdf:resource='#PolicyAction'/>
   <rdfs:range rdf:resource='#PolicyRule'/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID='SystemHasPolicyRule'>
  <rdf:type rdf:resource='#WeakProperty'/>
  <rdfs:domain rdf:resource='ccm:System'/>
  <rdfs:range rdf:resource='#PolicyRule'/>
  <owl:inverseOf rdf:resource='#PolicyRuleInSystem'/>
</owl:ObjectProperty>
<owl:Class rdf:about='ccm:System'>
  <rdfs:comment>Asociacion PolicyRuleInSystem. 0..*, Weak</rdfs:comment>
  <owl:Restriction>
    <owl:onProperty rdf:resource='#SystemHasPolicyRule'/>
    <owl:allValuesFrom rdf:resource='#PolicyRule'/>
  </owl:Restriction>
</owl:Class>
```

Se observa cómo ha sido necesario para ello definir una propiedad *SystemHasPolicyRule* inversa a la propiedad *PolicyRuleInSystem* y de tipo *WeakProperty*. Así como asociar esta propiedad a la clase *System*, lo cual, de nuevo, es posible pese a que esta clase no esté en el mismo esquema CIM, y por tanto no esté en la misma ontología OWL (uso de *rdf:about*). Del mismo modo se ha definido la propiedad *PolicyActionInPolicyRule* inversa a *PolicyRuleAggregatesPolicyGroup*.

OWL permite también explicitar características implícitas en las asociaciones CIM. Por ejemplo, el carácter transitivo de la agregación *PolicySetComponent* queda reflejado en la siguiente especificación OWL:

```
<owl:TransitiveProperty rdf:ID="PolicySetComponent">
   <rdfs:domain rdf:resource="#PolicySet"/>
   <rdfs:range rdf:resource="#PolicySet"/>
</owl:TransitiveProperty>
<owl:TransitiveProperty rdf:ID="PolicySetAggregation">
   <rdf:type rdf:resource="#AggregationProperty"/>
   <rdfs:domain rdf:resource="#PolicySet"/>
   <rdfs:range rdf:resource="#PolicySet"/>
   <owl:inverseOf rdf:resource="#PolicySetComponent"/>
</owl:TransitiveProperty>
```

3.3.6.2. "Mapping" de asociaciones n-arias (n>2)

Cada asociación CIM no binaria de tipo direccional se mapea a una clase OWL y cada referencia (salvo la referencia origen según el sentido/dirección de la asociación) se mapea a una propiedad *ObjectProperty* cuyo dominio es dicha clase OWL y cuyo rango es la clase CIM ligada a dicha referencia. La referencia origen se mapea a una propiedad *ObjectProperty* cuyo dominio es la clase CIM ligada a dicha referencia y cuyo rango es la clase OWL creada. Además, se precisan subclases de Restrict que asocien las propiedades OWL a cada clase OWL creada para la referencia origen y a la clase OWL que mapea la asociación, respectivamente, y que restrinja sus cardinalidades.

3.3.6.3. "Mapping" del concepto de sobreescritura

La propiedades CIM presentan asociaciones "reflexivas" cuya semántica incluye el concepto de sobreescritura en el meta-modelo. Estas asociaciones se mapean estableciendo propiedades rdfs:subPropertyOf entre las propiedades OWL que mapean a su vez las propiedades CIM implicadas. Del mismo modo, las asociaciones CIM se pueden organizar en jerarquías de herencia. Estas relaciones se mapean estableciento propiedades *rdfs:subPropertyOf* entre las propiedades OWL que mapean a su vez las asociaciones CIM implicadas. A modo de ejemplo, se muestra a continuación la especificación OWL de todas las relaciones de herencia entre las asociaciones CIM presentes en el modelo *CIM Core Policy Model*. La figura 3.3 ilustra esta jerarquía de herencia:

```
<owl:ObjectProperty rdf:about='PolicyRuleInSystem'>
  <rdfs:subPropertyOf rdf:resource='PolicySetSystem'/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about='PolicyGroupInSystem'>
  <rdfs:subPropertyOf rdf:resource='PolicySetSystem'/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about='PolicySetInSystem'>
  <rdfs:subPropertyOf rdf:resource='PolicyInSystem'/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about='ReusablePolicy'>
  <rdfs:subPropertyOf rdf:resource='PolicyInSystem'/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about='PolicyInSystem'>
  <rdfs:subPropertyOf rdf:resource='ccm:Dependency'/>
</owl:ObjectProperty>
```

3.3.6.4. Ejemplo del poder expresivo del "mapping" a OWL

El "mapping" del concepto de sobreescritura de las asociaciones CIM no sólo consigue salvaguardar la semántica, sino que permite hacer explícitas ciertas pro-

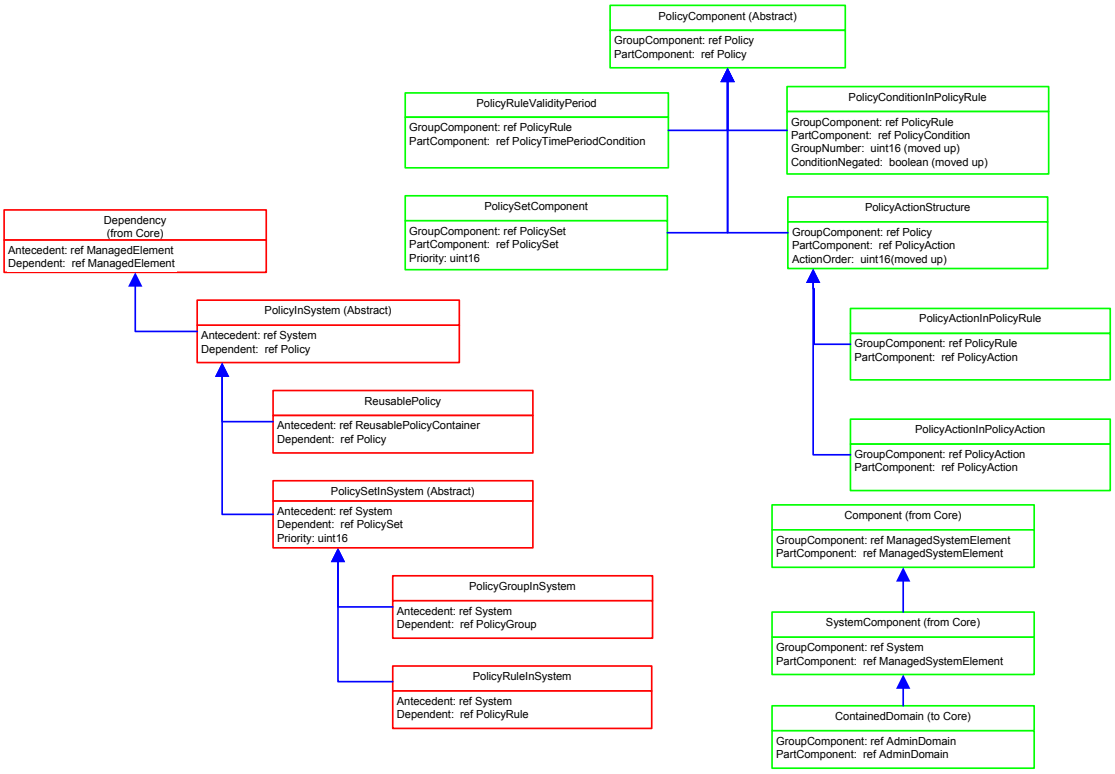

Figura 3.3: Jerarquía de herencia de las clases asociación presentes en el modelo de políticas CIM como instancias de la meta-clase ASSOCIATION (DMTF Service Level Agreements WG)

piedades que el meta-modelo CIM tan sólo podía expresar de manera textual. Así, siguiendo con el ejemplo propuesto, el modelo de políticas CIM incluye al menos dos elementos que muestran la incapacidad en CIM para expresar restricciones complejas que implican múltiples asociaciones: el modelado de las condiciones de una política especificas para una regla y el modelado de las acciones implicadas por una política.

Tomando las clases *PolicyCondition* y *PolicyRule* como ejemplo, se observa que la asociación *PolicyConditionInPolicyRule* tiene cardinalidad 0..* en ambos extremos. Globalmente, estas cardinalidades son correctas. Sin embargo, si se examina estas cardinalidades por separado para el caso de una condición específica para una regla y el caso de una condición reutilizable, se observa que:

- Para una *PolicyCondition* específica de una regla, la cardinalidad de *PolicyConditionInPolicyRule* en el extremo *PolicyRule* es 1..1, en lugar de 0..*, puesto que la condición es única para una regla.

- Para una *PolicyCondition* reutilizable, sin embargo, la cardinalidad de *Policy-*

ConditionInPolicyRule en el extremo *PolicyRule* es 0..*, por lo que la asociación debe recoger este caso más general. Este caso es más importante, ya que la condición puede entonces estar asociada con cero o más reglas, no pudiendo tomar entonces los valores a través de dicha asociación. Sólo en el caso de una condición específica esta cardinalidad será 1..1, por lo que podría tomar los valores de las claves a través de dicha asociación, que en este caso soportaría su propagación manual. No obstante, de nuevo el modelo no recoge este hecho, ya que la asociación debe reflejar el caso más general, esto es, 0..*.

El "mapping" a OWL permite sin embargo expresar explícitamente esta situación. El problema deriva del hecho de que las asociaciones UML no son objetos de primera clase: ligan explícitamente dos clases, por lo que están estableciendo tanto restricciones de dominio como restricciones de rango. Además, si P es una subclase de Q, entonces $\forall x, P(x) \rightarrow Q(x)$. Luego, si se definió la clase Q de modo que todas sus instancias tuvieran una asociación A_q, implícitamente se estaría definiendo que todas las instancias de P tienen la asociación A_q (incluidas sus cardinalidades). Sin embargo, si se define la clase P de modo que todas sus instancias presenten la asociación A_p, entonces aquellas instancias de Q que no sean instancias de P (instancias directas de Q) no podrán en ningún caso tener la asociación A_p. OWL, sin embargo, trata las *ObjectProperty* como objetos de primera clase, por lo que permite que en su definición se establezca la cardinalidad menos restrictiva y que, posteriormente, se restrinja esta cardinalidad a la hora de ligar esta propiedad a diferentes subconjuntos de elementos de una misma clase. Se muestra a continuación la especificación OWL resultante del "mapping" de esta propiedad:

```
<owl:ObjectProperty rdf:ID="PolicyConditionInPolicyRule">
  <rdfs:comment>Asociacion PolicyConditionInPolicyRule. 0..*</rdfs:comment>
  <rdfs:domain rdf:resource="#PolicyCondition"/>
  <rdfs:range rdf:resource="#PolicyRule"/>
</owl:ObjectProperty>
<owl:Class rdf:about="PolicyCondition">
  <rdfs:comment>Asociacion PolicyConditionInPolicyRule. 0..*</rdfs:comment>
  <owl:Restriction>
    <owl:onProperty rdf:resource="#PolicyConditionInPolicyRule"/>
    <owl:allValuesFrom rdf:resource="#PolicyRule"/>
  </owl:Restriction>
</owl:Class>
<owl:Class rdf:about="RuleSpecificPolicyCondition">
  <rdfs:comment>Asociacion PolicyConditionInPolicyRule. 1..1</rdfs:comment>
  <owl:Restriction>
    <owl:onProperty rdf:resource="#PolicyConditionInPolicyRule"/>
    <owl:allValuesFrom rdf:resource="#PolicyRule"/>
  </owl:Restriction>
  <owl:Restriction>
    <owl:onProperty rdf:resource="#PolicyConditionInPolicyRule"/>
```

```
      <owl:cardinality>1</owl:cardinality>
    </owl:Restriction>
  </owl:Class>
```

Nótese que no resulta adecuado en este caso crear una jerarquía de propiedades mediante el axioma *subPropertyOf*, ya que el modelo original sólo contempla una asociación *PolicyConditionInPolicyRule* y no una jerarquía de estas, por lo que el "mapping" no respetaría el modelo original.

3.3.7. "Mapping" de dependencias CIM

Una *dependencia* CIM, al contrario de lo que sucede con las asociaciones CIM, sólo puede ser binaria y no tiene roles. Una dependencia se mapea directamente en una propiedad de objeto OWL (ObjectProperty). Como identificador y como etiqueta de esta propiedad se toma el nombre de la dependencia, su dominio coincidirá con el nombre del clasificador fuente y su rango con el nombre del clasificador destino.

3.3.8. "Mapping" de calificadores y propiedades de asociaciones

Las asociaciones CIM pueden incluir propiedades.

Por otra parte, en el meta-modelo CIM pueden distinguirse dos tipos de calificadores:

Calificadores de tipos de datos CIM : Algunos tipos de datos intrínsecos se suelen utilizar con semánticas bien definidas (e.g. counter, gauge, octetstring, ArrayType("Indexed") originando así pseudo-tipos (ya que no se tratan como tipos intrínsecos). En CIM no se asocia ninguna semántica formal a estos calificadores. Diferentes implementaciones pueden introducir calificadores arbitrarios que tan solo se contemplan con propósitos de documentación. De hecho, la especificación del modelo CIM no incluye ninguna relación de calificadores estándar.

Calificadores de NamedElement : Califican clases, asociaciones, indicaciones, métodos, parámetros, triggers, instancias, propiedades y referencias. Se utilizan para calificar estos elementos nombrados. Proporcionan un mecanismo de extensibilidad controlada al meta-modelo CIM, ya que añaden información semántica adicional a estos elementos. Todos los calificadores constan de un nombre, un tipo, un valor, un alcance, un comportamiento y un valor por

omisión. Algunos de estos calificadores son mutuamente exclusivos y el uso de alguno de ellos implica algunas restricciones en el valor de otro calificador.

Ambos tipos de calificadores, así como las propiedades de las asociaciones CIM se mapean mediante metapropiedades.

Se define una metapropiedad OWL como una propiedad acerca de otras propiedades, esto es, una propiedad cuyo dominio a su vez es otra [meta]propiedad. En la sección 3.3.1 se demostró la viabilidad de esta decisión en *OWL Full*.

Se muestra a continuación el "mapping" de las propiedades presentes en la asociación *PolicyConditionInPolicyRule*:

```
<owl:ObjectProperty rdf:about="PolicyConditionInPolicyRule">
  <owl:Restriction>
    <owl:onProperty rdf:resource="#GroupNumber"/>
    <owl:cardinality>1</owl:cardinality>
  </owl:Restriction>
  <owl:Restriction>
    <owl:onProperty rdf:resource="#ConditionNegated"/>
    <owl:cardinality>1</owl:cardinality>
  </owl:Restriction>
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:about="GroupNumber">
  <rdfs:comment>DatatypeProperty para la propiedad GroupNumber de la
                        propiedad PolicyConditionInPolicyRule</rdfs:comment>
  <rdf:type rdf:resource="owl:FunctionalProperty"/>
  <rdfs:domain rdf:resource="#PolicyConditionInPolicyRule"/>
  <rdfs:range rdf:resource="xsd:uint16"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="ConditionNegated">
  <rdfs:comment>DatatypeProperty para la propiedad ConditionNegated de la
                        propiedad PolicyConditionInPolicyRule</rdfs:comment>
  <rdf:type rdf:resource="owl:FunctionalProperty"/>
  <rdfs:domain rdf:resource="#PolicyConditionInPolicyRule"/>
  <rdfs:range rdf:resource="xsd:boolean"/>
</owl:DatatypeProperty>
```

El cuadro 3.9 resume el "mapping" presentado:

Las descripciones de los objetos gestionados en una especificación CIM-OWL puede validarse contra un espacio de nombres activo. Este proceso valida tanto la corrección sintáctica de la especificación como la corrección semántica contra una implementación concreta. En este último caso, el conocimiento intercambiado por dos agentes acerca de los objetos de gestión del entorno representa una instancia de la ontología y puede validarse contra ésta.

Metaelemento CIM	**Vocabulario OWL**	**Comentario/Axiomas**
Clase	owl:Class	Class@rdf:ID=Class.name
Atributo (tipo subclase de ManagedElement)	owl:FunctionalProperty	En la clase OWL que lo contiene Class/Restriction/ onProperty@cardinality=1
Atributo (tipo básico)	owl:DatatypeProperty	En la clase OWL que lo contiene Class/Restriction/ onProperty@cardinality=1 DatatypeProperty/ rdf:type=owl:FunctionalProperty
Atributo []	owl:DatatypeProperty | owl:ObjectProperty	En la clase OWL que lo contiene Class/Restriction/ onProperty@minCardinality=1
Atributo clave	rdf:type=owl:KeyProperty	Metapropiedad com:isKey=true
Asociación	owl:ObjectProperty	
Referencia directa	owl:ObjectProperty	samePropertyAs[asociación]
Referencia inversa	owl:ObjectProperty	inverseOf[rol directo] e inverseOf[asociación]
Cardinalidad 1 ó 0..1 en la referencia destino	owl:FunctionalProperty	Si 1..1, Class/Restriction/ onProperty@Cardinality=1
Cardinalidad x..y en la referencia destino	minCardinality=x, y=∞ | maxCardinality=y	
Asociación «transitive»	owl:TransitiveProperty	domain=range
Asociación «Symmetric»	owl:SymmetricProperty	domain=range
Agregación	owl:AggregationProperty	Metapropiedad com:isAggregation=true
Atributo en asociación	(Metapropiedad) owl:DatatypeProperty | ObjectProperty	[ObjectProperty | DatatypeProperty]/Restriction/ onProperty@cardinality=1
Dependencia	owl:DependencyProperty	Metapropiedad com:isDependency=true
Generalización	owl:subClassOf	
Asociación múltiple		
Clase asociación	Metapropiedad DatatypeProperty cuyo dominio sea la ObjectProperty que mapea la asociación	
Asociación n-aria (n>2)	owl:Class y n ObjectProperty	
Restricción disjoint	owl:disjointWith	
Objeto		
Tipo primitivo	Tipo XML Schema	
Clase enumeración	class@owl:oneOf	

Cuadro 3.9: Resumen del "mapping" CIM-OWL FULL

3.4. Adecuación del "Mapping" CIM-OWL a la Lógica Descriptiva

El "mapping" CIM-OWL descrito en la sección anterior aprovecha al máximo las capacidades de modelado del lenguaje de ontologías CIM. Se trata por tanto de un "mapping" de CIM a OWL-FULL que utiliza algunas características satisfechas tan solo por esta especificación total de OWL y que viola determinadas restricciones de los razonadores basados en lógicas descriptivas [HST99, HM01], por lo que es poco probable que un sistema de razonamiento automático soporte razonamiento de tipo completo para todas sus características. En esta sección se describe un "mapping" más restrictivo de CIM a OWL-DL específico para la utilización de sistemas de razonamiento automático basados en lógicas descriptivas [McG03].

El cuadro 3.10 recoge los constructores de conceptos y roles utilizados en el desarrollo del "mapping" que se propone, con el fin de configurar un tipo de lógica descriptiva con expresividad suficiente. También muestra la complejidad asociada a sus dos principales servicios: subsunción ($\models C \sqsubseteq D$) y chequeo de instancia ($\models C(i)$). En dicho cuadro se observa que ha sido preciso utilizar una lógica $\mathcal{AL}\varepsilon\mathcal{CNOQ}^{-}_{\mathcal{HR}^{+}\circ}$ [2]. Esta decisión supone un compromiso entre la expresividad del lenguaje utilizado en la construcción de las bases de conocimiento terminológico (TBox) y la complejidad implicada en los procesos de razonamiento tanto sobre las bases de conocimiento terminológico, como sobre las bases de conocimiento asertivo (ABox). En este sentido, la utilización de otros tipos de lógica descriptiva como puede ser la familia de lógicas derivadas de la lógica $\mathcal{DLR}$, que eliminan la restricción de roles binarios e introduce constructores para roles n-arios, hubiese permitido la elaboración de un "mapping" más intuitivo que el propuesto en esta tesis, pero a costa de un coste computacional mucho mayor[3].

Al subconjunto $\mathcal{ALCQ}^{-}_{\mathcal{H}R^{+}}$ de la lógica utilizada se le suele denominar $\mathcal{SHIQ}$, sin embargo se ha decidido no partir de esta abreviación para evitar confusiones

[2] El esquema de nombres utilizado para denotar los diferentes tipos de lógicas descriptivas ha sido tomado de [McG03] y consiste en asignar una letra o símbolo a cada extensión expresiva de la lógica elemental $\mathcal{AL}$. Básicamente, estas letras y símbolos se escriben (1) a continuación de $\mathcal{AL}$ si se trata de la inclusión de un constructor de conceptos, (2) como un superíndice si se trata de un constructor de rol, y (3) mediante un subíndice si se trata una restricción en la interpretación de los roles.

[3] Las pruebas realizadas en la fase de obtención de resultados mostraron que la herramienta RACER [HM01] clasificaba en pocos segundos una base de conocimiento TBox con la totalidad de los modelos CIM elaborados por el IETF en su versión 2.6, expresados en lógica descriptiva siguiendo el "mapping" propuesto, mientras que esa misma herramienta era incapaz de clasificar la base de conocimiento expresada en $\mathcal{DLR}_{reg}$, esto es, la extensión de la lógica $\mathcal{DLR}$ con los constructores de unión, composición y cierre transitivo de roles binarios como proyección de los roles n-arios sobre dos de sus componentes

a la hora de denotar el resto de construcciones necesarias (composición de roles, negación de conceptos y cardinalidad no calificada).

Por otra parte, la lógica escogida es realmente un superconjunto de la lógica mínima necesaria para realizar el "mapping", ya que se tienen las siguientes equivalencias:

$$C \sqcup D \equiv \neg(\neg C \sqcap \neg D) \text{ y}$$
$$\exists R.C \equiv \neg\forall R\neg C$$

que hacen que $\mathcal{C} \equiv \mathcal{U}\varepsilon$. Sin embargo, se ha optado por no utilizar estas simplificaciones con el fin de aportar mayor claridad al "mapping" sin introducir con ello una mayor complejidad de cómputo, ya que ambas lógicas tienen la misma expresividad.

La tabla 3.10 no recoge los axiomas permitidos en la elaboración de las bases de conocimiento TBox. Además de los axiomas tradicionales de subsunción de conceptos ($C \sqsubseteq D$) y equivalencia de conceptos ($C \equiv D$), restringidos en el sentido de que sólo D puede ser una expresión de concepto (y por tanto C debe ser un concepto atómico), se ha utilizado también el axioma de subsunción de roles con el fin de poder crear jerarquías de roles. De ahí el subíndice $\mathcal{H}$.

Al contrario de lo que ocurre con el "mapping" original, que ha sido descrito en base a constructores y axiomas OWL-FULL, este nuevo "mapping" se describe a nivel de construcciones y axiomas $\mathcal{AL}\varepsilon\mathcal{CNOQ}^{-}_{\mathcal{H}R^{+}\circ}$ con el fin de hacer explícito cómo esta lógica descriptiva captura la semántica de los diferentes elementos CIM "mapeados". El cuadro 3.11 recoge la correspondencia existente entre las diferentes construcciones y axiomas $\mathcal{AL}\varepsilon\mathcal{CNOQ}^{-}_{\mathcal{H}R^{+}\circ}$ utilizados en el "mapping" y la sintaxis OWL-DL. A partir de este cuadro se puede obtener directamente la expresión OWL-DL de cualquier descripción en lógica descriptiva de una conceptualización CIM.

La semántica formal del "mapping" del meta-esquema CIM a $\mathcal{AL}\varepsilon\mathcal{CNOQ}^{-}_{\mathcal{H}R^{+}\circ}$ se basa en una teoría de modelos en la que $\mathcal{I} = \langle D, \cdot^{\mathcal{I}} \rangle$ es una interpretación en la que:

$D \neq \emptyset$ representa un dominio o universo de discurso tal que $D = \Sigma \cup \Upsilon$ con $\Upsilon = \bigcup_{i=1}^{n} \Upsilon_{D_i}$, $\Upsilon_{D_i} \cap \Upsilon_{D_j} = \varnothing$, y $\Sigma \cap \Upsilon = \varnothing$, tal que Σ es el dominio de objetos gestionados y Υ_{D_i} es el conjunto de valores asociados con cada tipo básico de datos D_i soportado por CIM (integer, string, etc).

$\cdot^{\mathcal{I}}$ es la función de interpretación que proyecta:

- $D_i^{\mathcal{I}} = \Upsilon_{D_i}$.
- $C_i^{\mathcal{I}} \subseteq \Sigma$.

Constructor	**Sintaxis**	**Semántica**	**Tipo de lógica**	**Compl.** $\models C \sqsubseteq D$	**Compl.** $\models C(i)$
Concepto atómico	A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$	$\mathcal{FL}_0$	P	P
Dominio	$\top$	$\Delta^{\mathcal{I}}$			
Vacio	$\bot$	$\emptyset$			
Conjunción	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$			
Universal	$\forall R.C$	$\{x \mid \forall y : R^{\mathcal{I}}(x,y) \rightarrow C^{\mathcal{I}}(y)\}$			
Existencial	$\exists R.\top$	$\{x \mid \exists y : R^{\mathcal{I}}(x,y)\}$	$\mathcal{FL}^-$	P	P
Negación Atómica	$\neg A$	$\Delta^{\mathcal{I}} \backslash A^{\mathcal{I}}$	$\mathcal{AL}$	P	P
Existencial Calificado	$\exists R.C$	$\{x \mid \exists y : R^{\mathcal{I}}(x,y) \wedge C^{\mathcal{I}}(y)\}$	$\mathcal{E}$	NP	PSPACE
Negación	$\neg C$	$\Delta^{\mathcal{I}} \backslash C^{\mathcal{I}}$	$\mathcal{C}$	PSPACE	PSPACE?
Enumeración	$a_1 \cdots a_n$	$a_1^{\mathcal{I}} \cdots a_n^{\mathcal{I}}$	$\mathcal{O}$	PSPACE	PSPACE
Disyunción	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$	$\mathcal{U}$		
Cardinalidad	$\geq nR$	$\{x \mid \sharp\{y \mid R^{\mathcal{I}}(x,y)\} \geq n\}$	$\mathcal{N}$		
	$\leq nR$	$\{x \mid \sharp\{y \mid R^{\mathcal{I}}(x,y)\} \leq n\}$			
	$= nR$	$\{x \mid \sharp\{y \mid R^{\mathcal{I}}(x,y)\} = n\}$			
Cardinalidad Calificada	$\geq nR.C$	$\{x \mid \sharp\{y \mid R^{\mathcal{I}}(x,y) \wedge C^{\mathcal{I}}(y)\} \geq n\}$	$\mathcal{Q}$	EXP	EXP
	$\leq nR.C$	$\{x \mid \sharp\{y \mid R^{\mathcal{I}}(x,y) \wedge C^{\mathcal{I}}(y)\} \leq n\}$			
	$= nR.C$	$\{x \mid \sharp\{y \mid R^{\mathcal{I}}(x,y) \wedge C^{\mathcal{I}}(y)\} = n\}$			
Selección	$f : C$	$\{x \in Dom(f^{\mathcal{I}}) \mid C^{\mathcal{I}}(f^{\mathcal{I}}(x))\}$	$R_{\mathcal{F}}$		
R. Transit.	R^+	$\bigcup_{n \geq 1}(R^{\mathcal{I}})^n$	$()^+$		
R. Inversos	R^-	$\{(y,x) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid R^{\mathcal{I}}(a,b)\}$	$()_{R^-}$		
Composicion de roles	$R \circ S$	$R^{\mathcal{I}} \circ S^{\mathcal{I}} = \{(x,z) \mid \exists y \in \Delta^{\mathcal{I}} \cdot R^{\mathcal{I}}(x,y) \wedge S^{\mathcal{I}}(y,z)\}$	$()_{R^\circ}$		

Cuadro 3.10: Descripción de los tipos de Lógicas Descriptivas utilizados y sus complejidades asociadas

Sintaxis OWL-DL	**Sintaxis** $\mathcal{AL}\varepsilon\mathcal{CNOQ}^{-}_{\mathcal{H}R^{+}\circ}$
complementOf (c)	$\neg C$
unionOf($c_1 \cdots c_n$)	$C_1 \sqcup \cdots \sqcup C_n$
intersectionOf($c_1 \cdots c_n$)	$C_1 \sqcap \cdots \sqcap C_n$
restriction(p allValuesFrom(c))	$\forall P.C$
restriction(p someValuesFrom(c))	$\exists P.C$
restriction(p minCardinality(n))	$\langle \geq nP \rangle$
restriction(p maxCardinality(n))	$\langle \leq nP \rangle$
restriction(p Cardinality(n))	$\langle = nP \rangle$
FunctionalProperty(p)	$\langle \leq 1P \rangle$
InverseFunctionalProperty(p)	$\langle \leq 1P^{-} \rangle$
domain(p,c)	$\forall P^{-}.C$
restriction(p range(c))	$\forall P.C$
subClassOf(c_1, c_2)	$C_1 \sqsubseteq C_2$
subPropertyOf(p,q)	$P \sqsubseteq Q$
inverseOf(p,q)	$P \equiv Q^{-}$
SymmetricProperty(p)	$P \equiv P^{-}$
TransitiveProperty(p)	$P \equiv P \circ^{i} P$,i=1..n
sameClassAs(c_1, c_2)	$C_1 \equiv C_2$
samePropertyAs(p,q)	$P \equiv Q$
oneOf(c,$c_1 \cdots c_n$)	$C \sqsubseteq C_1 \sqcup \cdots \sqcup C_n$
disjointWith(c,$c_1 \cdots c_n$)	$C \sqsubseteq \neg C_1 \sqcap \cdots \sqcap \neg C_n$
inverseOf(p,q)	$P \equiv Q^{-}$
equivalentProperty(p,q)	$P \equiv Q$

Cuadro 3.11: Correspondencia entre la sintaxis OWL y $\mathcal{AL}\varepsilon\mathcal{CNOQ}^{-}_{\mathcal{H}R^{+}\circ}$

- $A_i^{\mathcal{I}} \subseteq \Sigma \times \Upsilon$.
- $R_i^{\mathcal{I}} \subseteq \Sigma \times \cdots \times \Sigma \equiv \Sigma^n$.

Con el fin de ilustrar el "mapping" propuesto, se describe de forma incremental parte del proceso de traducción del *Modelo Central* de CIM (*CIM Core Model*) mostrado en las figuras 3.4 y 3.5. El apéndice A recoge el "mapping" completo.

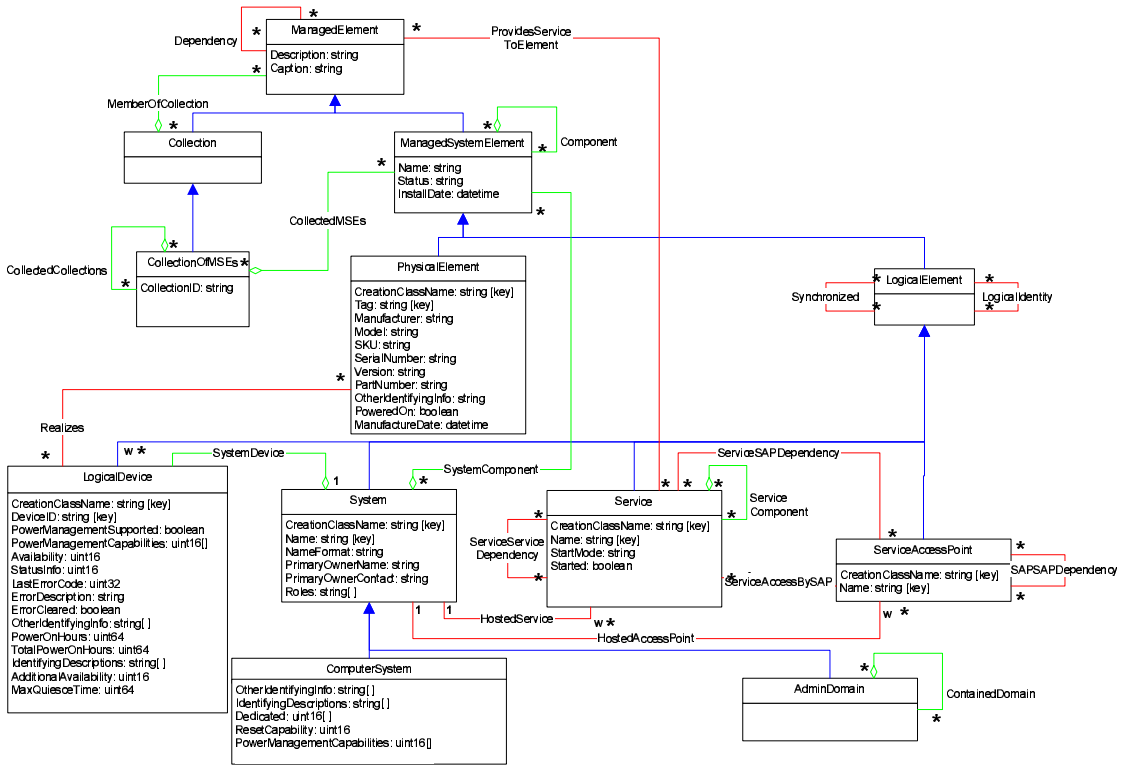

Figura 3.4: Modelo Central de CIM versión 2.6 (DMTF SysDev-WG)

3.4.1. "Mapping" de clases CIM

Una clase CIM denota un conjunto de objetos con características comunes en términos de propiedades y asociaciones, por lo que se representa como un concepto C en lógica descriptiva $\mathcal{ALECNOQ}^{-}_{\mathcal{HR}^{+}\circ}$.

3.4.2. "Mapping" de jerarquías de generalización

Una relación de generalización o herencia entre dos clases CIM especifica que cada instancia de la clase *hija* es también una instancia de la clase *padre* y que las instancias de la clase hija heredan las propiedades presentes en la clase padre (y

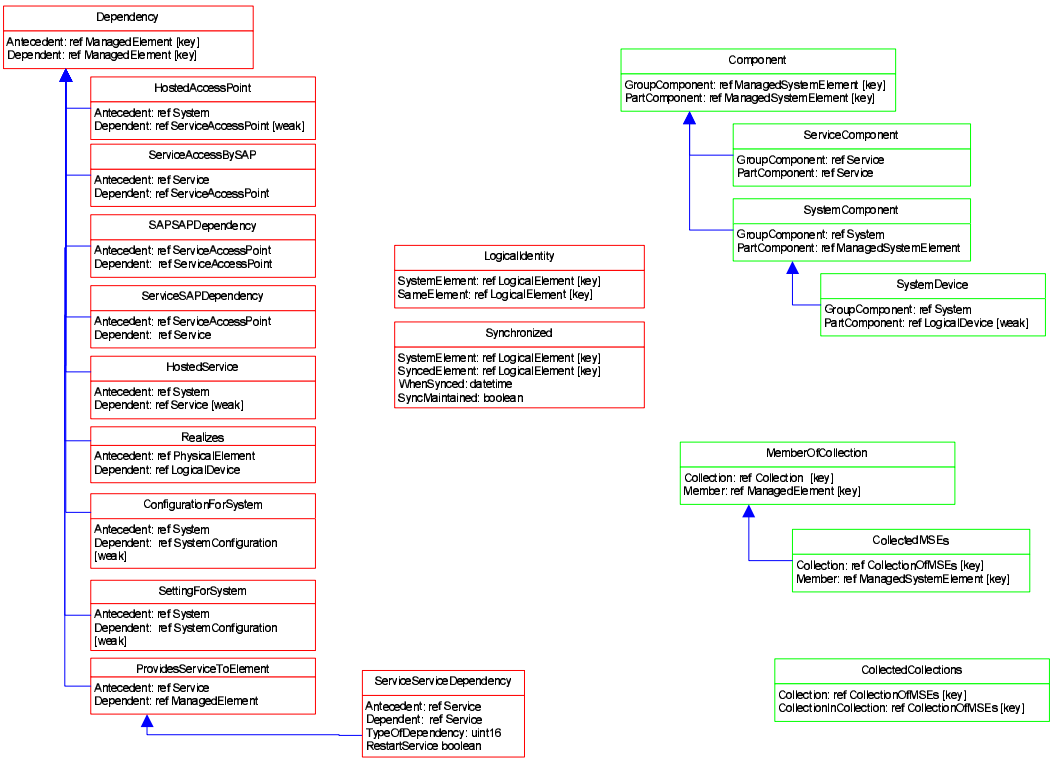

Figura 3.5: Jerarquía de herencia de las clases asociación presentes en el *Modelo Central* de CIM como instancias de la meta-clase ASSOCIATION (DMTF SysDev-WG)

satisfacen otras adicionales) y pueden participar en sus asociaciones. La relación de generalización CIM se expresa como una herencia entre conceptos $\mathcal{AL}\varepsilon\mathcal{CNOQ}^{-}_{\mathcal{HR}^{+}\circ}$ aprovechando que la semántica de las aserciones de inclusión ($C_i \sqsubseteq C_j$) está basada en la inclusión de conjuntos y respeta el concepto de sustitución.

La relación de generalización CIM permite distinguir cuatro tipos de situaciones:

Parcial y no disjunta . El significado de esta restricción es:

$C_i^{\mathcal{I}} \subseteq C^{\mathcal{I}}, i = 1, ..., n.$

que puede traducirse a un conjunto de fórmulas en lógica de primer orden:

$\forall x \cdot C_i(x) \rightarrow C(x), i = 1, ..., n.$

Esta restricción puede expresarse en lógica descriptiva $\mathcal{AL}\varepsilon\mathcal{CNOQ}^{-}_{\mathcal{HR}^{+}\circ}$ como:

$C_i \sqsubseteq C, i = 1, ..., n.$

Parcial y disjunta . El significado de esta restricción es:

$C_i^{\mathcal{I}} \subseteq C^{\mathcal{I}}, i = 1, ..., n.$

$C_i^{\mathcal{I}} \cap C_j^{\mathcal{I}} = \varnothing, i = 1, ..., n.$

que puede traducirse a un conjunto de fórmulas en lógica de primer orden:

$\forall x \cdot C_i(x) \rightarrow C(x) \wedge \bigwedge_{j=i+1}^{n} \neg C_j(x), i = 1, ..., n.$

Esta restricción puede expresarse en lógica descriptiva $\mathcal{AL}\varepsilon\mathcal{CNOQ}^{-}_{\mathcal{H}R^{+}\circ}$ como:

$C_i \sqsubseteq C, i = 1, ..., n.$

$C_i \sqsubseteq \neg C_j$, para todo $i \neq j$.

Total y no disjunta El significado de esta restricción es:

$C_i^{\mathcal{I}} \subseteq C^{\mathcal{I}}, i = 1, ..., n.$

$C^{\mathcal{I}} \subseteq \bigcup_{i=1}^{n} C_i^{\mathcal{I}}$

que puede traducirse a un conjunto de fórmulas en lógica de primer orden:

$\forall x \cdot C_i(x) \rightarrow C(x), i = 1, ..., n.$

$\forall x \cdot C(x) \rightarrow \bigvee_{i=1}^{n} C_i(x)$

Esta restricción puede expresarse en lógica descriptiva $\mathcal{AL}\varepsilon\mathcal{CNOQ}^{-}_{\mathcal{H}R^{+}\circ}$ como:

$C_i \sqsubseteq C, i = 1, ..., n.$

$C \sqsubseteq \bigsqcup_{i=1}^{n} C_i$

Total y disjunta . Este es el tipo menos utilizado por el carácter abierto intrínseco a las ontologías. Sin embargo, debe considerarse su presencia al ser el tipo que aporta más semántica al modelo. El significado de esta restricción es:

$C_i^{\mathcal{I}} \subseteq C^{\mathcal{I}}, i = 1, ..., n.$

$C_i^{\mathcal{I}} \cap C_j^{\mathcal{I}} = \varnothing$, para todo $i \neq j$

$C^{\mathcal{I}} \subseteq \bigcup_{i=1}^{n} C_i^{\mathcal{I}}$

que puede traducirse a un conjunto de fórmulas en lógica de primer orden:

$\forall x \cdot C_i(x) \rightarrow \bigvee_{i=1}^{n} C_i(x)$

$\forall x \cdot C_i(x) \rightarrow C(x) \wedge \bigwedge_{j=i+1}^{n} \neg C_j(x)$

Esta restricción puede expresarse en lógica descriptiva $\mathcal{AL}\varepsilon\mathcal{CNOQ}^{-}_{\mathcal{H}R^{+}\circ}$ como:

$C_i \sqsubseteq C, i = 1, ..., n.$

$C_i \sqsubseteq \neg C_j$, para todo $i \neq j$.

$C \sqsubseteq \bigsqcup_{i=1}^{n} C_i$

Esta formalización captura también la relación de generalización entre [clases] asociaciones CIM.

Se muestra a continuación un ejemplo de formalización de la relación de generalización total y disjunta existente entre los conceptos *ManagedSystemElement*, *PhysicalElement* y *LogicalElement*. El meta-esquema CIM no permite expresar formalmente los tipos de relaciones de generalización descritos, si bien es costumbre indicarlos gráficamente en el diagrama haciendo uso de la notación UML.

$$ManagedSystemElement \sqsubseteq Class \sqcap (PhysicalElement \sqcup LogicalElement)$$
$$PhysicalElement \sqsubseteq Class \sqcap ManagedSystemElement \sqcap \neg LogicalElement$$
$$LogicalElement \sqsubseteq Class \sqcap ManagedSystemElement \sqcap \neg PhysicalElement$$

3.4.3. "Mapping" de propiedades CIM

La restricción impuesta por la asignación de un atributo A con tipo de datos D a una clase C es:

$$C^{\mathcal{I}} \subseteq \{x \in \Sigma \mid \sharp(A^{\mathcal{I}} \cap (\{x\} \times \Upsilon_D)) \geq 1\}$$

que puede traducirse a una fórmula en lógica de primer orden:

$$\forall x \cdot C(x) \rightarrow \exists y \cdot A(x, y) \wedge D(y)$$

Esta restricción puede expresarse en lógica descriptiva $\mathcal{AL}\varepsilon\mathcal{CNO}\mathcal{Q}^{-}_{\mathcal{H}R^{+\circ}}$ como:

$C \sqsubseteq \langle = 1A \rangle \sqcap \exists A.D$ para atributos con un único valor,

$C \sqsubseteq \langle \geq 1A \rangle \sqcap \forall A.D$ para atributos de tipo A[], y

$C \sqsubseteq \langle \geq n_i A \rangle \sqcap \langle \leq n_j A \rangle \sqcap \forall A.D$ para atributos con cardinalidad $(n_i..n_j)$.

siendo C un concepto, A un rol binario y D un tipo de datos.

El "mapping" $CIM - OWL_{FULL}$ permite expresar la semántica tanto de claves simples como de claves compuestas gracias a la intersección no vacía entre los conjuntos de conceptos y roles. Sin embargo, el "mapping" de CIM a lógica descriptiva desarrollado tan solo permite expresar la semántica de atributos actuando como claves simples:

$$C \sqsubseteq \langle = 1A \rangle \sqcap \exists A.D \sqcap \langle \leq 1A^{-} \rangle$$

La semántica de una clave compuesta viene definida por una relación

$$\mathcal{R} : C \rightarrow (A_1 \times A_2 \times \cdots \times A_n)$$

tal que $\mathcal{R}$ es inyectiva y $\mathcal{R}^-$ es funcional, por lo que sería preciso poder considerar la relación $(A_1 \times A_2 \times \cdots \times A_n)$ como un concepto de modo que sus instancias pudiesen constituir el rango de la relación $\mathcal{R}$.

Se muestra a continuación el "mapping" de los atributos de la clase *System*. Todos ellos son atributos únicos de tipo *String*, salvo *Roles* que es un atributo multivaluado.

$$
\begin{aligned}
System \sqsubseteq & Class \sqcap LogicalElement \sqcap \\
& \exists CreationClassName.string \sqcap \langle \leq 1 CreationClassName \rangle \sqcap \\
& \exists Name.string \sqcap \langle \leq 1 Name \rangle \sqcap \\
& \exists NameFormat.string \sqcap \langle \leq 1 NameFormat \rangle \sqcap \\
& \exists PrimaryOwnerName.string \sqcap \langle \leq 1 PrimaryOwnerName \rangle \sqcap \\
& \exists PrimaryOwnerContact.string \sqcap \langle \leq 1 PrimaryOwnerContact \rangle \sqcap \\
& \forall Roles.string \sqcap \langle \geq 1 Roles \rangle
\end{aligned}
$$

3.4.4. "Mapping" de asociaciones y dependencias CIM

La restricción impuesta por la interrelación de n clases $C_1 \cdots C_n$ mediante una asociación R es:

$$R^{\mathcal{I}} \subseteq C_1^{\mathcal{I}} \times \cdots \times C_n^{\mathcal{I}}$$

que puede traducirse a una fórmula en lógica de primer orden:

$$\forall x_1, ..., x_n \cdot R(x_1, ..., x_n) \rightarrow C_1(x) \wedge \cdots \wedge C_n(x)$$

Esta restricción puede expresarse en lógica descriptiva mediante un rol n-ario o bien a través de la transformación de la asociación R en un concepto A y n roles $r_1 ... r_n$. Esta última opción permite representar fácilmente propiedades y calificadores asociados a dicha asociación mediante nuevos roles tal y como se ha descrito en la sección 3.4.3. Además, esta última opción sigue la semántica CIM que determina que las asociaciones no deben manejarse como relaciones inversas con referencias asociadas a cada clase participante, sino como un objeto diferente que tiene asociadas referencias a las clases participantes.

$$A \sqsubseteq \exists r_1.C_1 \sqcap \cdots \sqcap \exists r_n.C_n \sqcap \langle \leq 1 r_1 \rangle \sqcap \cdots \sqcap \langle \leq 1 r_n \rangle$$

Puesto que una asociación CIM es una especialización de una clase CIM, pueden existir, además de propiedades cuyo dominio sea una asociación, relaciones de generalización entre asociaciones. Éstas últimas se tratan como relaciones de herencia entre conceptos.

Una *dependencia* CIM, al contrario de lo que sucede con las asociaciones CIM, sólo puede ser binaria y no tiene roles. Una dependencia puede expresarse pues mediante un rol binario.

3.4.4.1. Expresión de las cardinalidades

CIM permite asociar cardinalidades a los roles de la asociación. Esto supone una nueva restricción por cada rol que puede expresarse como:

$$C_i^{\mathcal{I}} \subseteq \{x \in \Sigma \mid m_i \leq \sharp(R^{\mathcal{I}} \cap (\Sigma \times \{x\} \times \Sigma)) \geq n_i\} \text{ i=1,...,n.}$$

que puede traducirse a una fórmula en lógica de primer orden:

$$\begin{aligned}\forall x_i \cdot C(x_i) \rightarrow & \exists^{\geq p} x_1, ..., x_{i-1}, x_{i+1}, ..., x_n \cdot R(x_1, ..., x_n) \wedge \\ & \exists^{\leq p} x_1, ..., x_{i-1}, x_{i+1}, ..., xn \cdot R(x_1, ..., x_n)\end{aligned}$$

con

$$\begin{aligned}&\exists^{\leq n} x \cdot R(x, y) \equiv \\ &\quad \forall x_1, ..., x_n, x_{n+1} \cdot R(x_1, y) \wedge \cdots \wedge R(x_n, y) \wedge R(x_{,n+1}, y) \rightarrow \\ &\quad (x_1 = x_2) \vee \cdots \vee (x_1 = x_n) \vee (x_1 = x_{n+1}) \vee \\ &\quad (x_2 = x_3) \vee \cdots \vee (x_2 = x_n) \vee (x_2 = x_{n+1}) \vee \\ &\quad \cdots \vee (x_n = x_{n+1})\end{aligned}$$

y

$$\begin{aligned}\exists^{\geq n} x \cdot R(x, y) \equiv & \exists x_1, ..., x_n \cdot R(x_1, y) \wedge \cdots \wedge R(x_n, y) \wedge R(x_{n+1}, y) \wedge \\ & \neg(x_1 = x_2) \wedge \cdots \wedge \neg(x_1 = x_n) \wedge \\ & \neg(x_2 = x_3) \wedge \cdots \wedge \neg(x_2 = x_n) \wedge \\ & \cdots \wedge (x_{n-1} = x_n)\end{aligned}$$

Por tanto, una asociación n-aria con cardinalidades $(k_i..l_i)$ puede expresarse en lógica descriptiva $\mathcal{AL}\varepsilon\mathcal{CNOQ}^{-}_{\mathcal{H}R^{+}\circ}$ como:

$$A \sqsubseteq \exists r_1.C_1 \sqcap \cdots \sqcap \exists r_n.C_n \sqcap \langle \leq 1 r_1 \rangle \sqcap \cdots \sqcap \langle \leq 1 r_n \rangle$$

$$C_i \sqsubseteq \forall r_i^{-}.A \sqcap \langle \geq k_i r_i^{-} \rangle \sqcap \langle \leq l_i r_i^{-} \rangle, i = 1, ..., n$$

Nótese que, en general sólo pueden definirse cardinalidades para roles no transitivos y que no tengan a su vez subroles transitivos. Esta no es una restricción intrínseca a la lógica descriptiva utilizada, sino de los sistemas de razonamiento deductivo asociados a las mismas.

3.4.4.2. Expresión de la navegabilidad de una asociación

En el caso de una asociación binaria, se introduce un nuevo rol A_r para restringir la cardinalidad en el concepto que actúa como dominio y explicitar la navegabilidad de la asociación.

$$A_r \equiv r_1^- \circ r_2$$
$$C_1 \sqsubseteq \forall A_r.C_2 \sqcap \langle \geq m_i A_r \rangle \sqcap \langle \leq m_j A_r \rangle$$
$$C_2 \sqsubseteq \forall A_r^-.C_2 \sqcap \langle \geq m_i A_r^- \rangle \sqcap \langle \leq m_j A_r \rangle$$
$$A \sqsubseteq \exists r_1.C_1 \sqcap \exists r_2.C_2 \sqcap \langle \leq 1 r_1 \rangle \sqcap \langle \leq 1 r_2 \rangle$$
$$C_1 \sqsubseteq \forall r_1^-.A \sqcap \langle \geq m_i r_1^- \rangle \sqcap \langle \leq m_j r_1^- \rangle$$
$$C_2 \sqsubseteq \forall r_2^-.A \sqcap \langle \geq n_i r_2^- \rangle \sqcap \langle \leq n_j r_2^- \rangle$$

El rol A_r permite además especificar la transitividad de una asociación (téngase en cuenta que r_1 y r_2 no son roles transitivos).

Se muestra a continuación el "mapping" de la agregación *ServiceComponent* y de la asociación binaria *HostedService* con cardinalidad (1..1) en su dominio y cardinalidad (0..*) en su rango. El ejemplo muestra también la utilización del rol *ServiceComponentRole* para especificar la navegabilidad de la asociación *ServiceComponent*. CIM especifica esta misma semántica de manera informal mediante una regla de nombrado para las referencias de la clase asociación correspondiente.

$$System \sqsubseteq \forall HostedService\_Antecedent^-.HostedService \sqcap \forall HostedServiceRole.Service \sqcap$$

$$Service \sqsubseteq \forall ServiceComponent\_Group^-.ServiceComponent \sqcap \forall ServiceComponent\_Part^-.ServiceComponent \sqcap \forall ServiceComponentRole.Service \sqcap \exists HostedService\_Dependent^-.HostedService \sqcap \langle \leq 1 HostedService\_Dependent^- \rangle \sqcap \exists HostedServiceRole^-.System \sqcap \langle \leq 1 HostedServiceRole \rangle$$

$$HostedService \sqsubseteq Association \sqcap \exists HostedService\_Antecedent.System \sqcap \langle \leq 1 HostedService\_Antecedent \rangle \sqcap \exists HostedService\_Dependent.Service \sqcap \langle \leq 1 HostedService\_Dependent \rangle$$

$$HostedServiceRole \equiv HostedService\_Antecedent^- \circ HostedService\_Dependent$$

$$ServiceComponent \sqsubseteq Aggregation \sqcap \exists ServiceComponent\_Group.Service \sqcap \langle \leq 1 ServiceComponent\_Group \rangle \sqcap \exists ServiceComponent\_Part.Service \sqcap \langle \leq 1 ServiceComponent\_Part \rangle$$

$$ServiceComponentRole \equiv ServiceComponent\_Group^{-} \circ ServiceComponent\_Part$$

El cuadro 3.12 sintetiza el "mapping" CIM-$\mathcal{AL}\varepsilon\mathcal{CNOQ}^{-}_{\mathcal{H}R^{+}\circ}$ descrito.

3.4.5. "Mapping" de calificadores

A la hora de "mapear" los calificadores CIM se ha seguido el siguiente criterio metodológico:

> Si la diferenciación entre dos conceptos tiene implicaciones concretas para sus relaciones con otros conceptos o supone restricciones para otras propiedades en otros conceptos, se crea un nuevo concepto. En otro caso se prefiere la utilización de una propiedad para expresar dicha diferenciación.

En el caso del meta-esquema CIM, los únicos calificadores que requieren la creación de nuevos conceptos son *Aggregation* y *Aggregate*. El resto pueden ser expresados mediante propiedades asociadas a los conceptos incluidos en el alcance del calificador. La expresión del Meta-esquema CIM mostrada en la siguiente sección recoge el "mapping" de los calificadores CIM según este criterio.

3.4.6. Expresión del Meta-esquema CIM en $\mathcal{AL}\varepsilon\mathcal{CNOQ}^{-}_{\mathcal{H}R^{+}\circ}$

Con el propósito de ilustrar el "mapping" de calificadores CIM propuesto en la sección anterior, se muestra a continuación la expresión del meta-esquema CIM en $\mathcal{AL}\varepsilon\mathcal{CNOQ}^{-}_{\mathcal{H}R^{+}\circ}$ según el proceso de traducción presentado en la sección anterior. Los conceptos y roles creados se utilizan como base a la hora de crear las representaciones $\mathcal{AL}\varepsilon\mathcal{CNOQ}^{-}_{\mathcal{H}R^{+}\circ}$ de otros esquemas CIM como el modelo de núcleo CIM de la siguiente sección o el modelo de políticas CIM presentado en el apéndice A. Esto facilitará la construcción de herramientas CASE avanzadas con sistemas de razonamiento incorporados que ayudarán a detectar inconsistencias. Un ejemplo es la detección de una relación de generalización incorrecta entre una clase y una [clase] asociación (La semántica del meta-esquema CIM aún no permite expresar esta restricción).

Elemento CIM	**Conceptos y roles introducidos**	**Axiomas DL añadido**
Clase C	Concepto C	
Atributo a de C con tipo T	Rol binario a	$C \sqsubseteq \langle = 1a \rangle \sqcap \exists a.T$
Atributo a de C act. como clave	Rol binario a	$C \sqsubseteq \langle = 1A \rangle \sqcap \exists A.D$ $\sqcap \langle \leq 1A^- \rangle$
Atributo a de C con tipo T[]	Rol binario a	$C \sqsubseteq \langle \geq 1a \rangle \sqcap \forall a.T$
Atributo a con card. asociada $(n_i..n_j)$	Rol binario a	$C \sqsubseteq \langle \geq n_i a \rangle \sqcap \langle \leq n_j a \rangle \sqcap \forall a.T$
Dependencia A	Rol binario A Roles R_1 y R_2	$\top \sqsubseteq \forall A.C_2 \sqcap \forall A^-.C_1$ $C_1 \sqsubseteq \forall A.C_2 \sqcap [\geq n_i A] \sqcap [\leq n_j A]$ $C_2 \sqsubseteq \forall A^-.C_1 \sqcap [\geq m_i A^-] \sqcap [\leq m_j A^-]$ $A \sqsubseteq R_1, R_1 \sqsubseteq A, A^- \sqsubseteq R_2, R_2 \sqsubseteq A^-$
Asociación n-aria con multiplicidad	Concepto A Roles $A_r, R_1 \dots R_n$	$A \sqsubseteq \exists R_1.C_1 \sqcap \dots \sqcap \exists R_n.C_n \sqcap$ $\langle \leq 1R_1 \rangle \sqcap \dots \sqcap \langle \leq 1R_n \rangle$ $C_i \sqsubseteq \forall R_i^-.A \sqcap \langle \geq n_i R_i^- \rangle \sqcap \langle \leq n_j R_i^- \rangle$ $i = 1, ..., n$
Asociación binaria con multiplicidad	Concepto A Roles A_r, R_1 y R_2	$A \sqsubseteq \exists R_1.C_1 \sqcap \exists R_2.C_2$ $\sqcap \langle \leq 1R_1 \rangle \sqcap \langle \leq 1R_2 \rangle$ $C_1 \sqsubseteq \forall R_1^-.A \sqcap \langle \geq m_i R_1^- \rangle \sqcap \langle \leq m_j R_1^- \rangle$ $C_2 \sqsubseteq \forall R_2^-.A \sqcap \langle \geq n_i R_2^- \rangle \sqcap \langle \leq n_j R_2^- \rangle$ $A_r \equiv R_1^- \circ R_2$ $C_1 \sqsubseteq \forall A_r.C_2 \sqcap \langle \geq m_i A_r \rangle \sqcap \langle \leq m_j A_r \rangle$ $C_2 \sqsubseteq \forall A_r^-.C_2 \sqcap \langle \geq m_i A_r^- \rangle \sqcap \langle \leq m_j A_r \rangle$
Relación de herencia parcial y no disjunta		$C_i \sqsubseteq C, i = 1, ..., n$
Relación de herencia parcial y disjunta		$C_i \sqsubseteq C, i = 1, ..., n$ $C_i \sqsubseteq \neg C$, para todo $i \neq j$
Relación de herencia total y no disjunta		$C_i \sqsubseteq C, i = 1, ..., n$ $C \sqsubseteq \bigsqcup_{i=1}^{n} C_i$
Relación de herencia total y disjunta		$C_i \sqsubseteq C, i = 1, ..., n$ $C_i \sqsubseteq \neg C$,para todo $i \neq j$ $C \sqsubseteq \bigsqcup_{i=1}^{n} C_i$

Cuadro 3.12: Síntesis del "mapping" CIM-$\mathcal{ALE}\mathcal{CNOQ}^-_{\mathcal{HR}^{+\circ}}$

$$\begin{aligned} NamedElement \sqsubseteq & \exists Name.String \sqcap \langle \leq 1 Name \rangle \sqcap \\ & \exists ElementSchema.Element^{-}.ElementSchema \sqcap \\ & \langle \leq 1 ElementSchema.Element^{-} \rangle \sqcap \\ & \forall Characteristics.Qualifier \sqcap \\ & \forall ElementTrigger\_Element^{-}.ElementTrigger \sqcap \\ & \forall ElementTriggerRole.Trigger \end{aligned}$$

$$\begin{aligned} ElementTrigger \sqsubseteq & \exists ElementTrigger\_Element.NamedElement \sqcap \\ & \exists ElementTrigger\_Trigger.Trigger \sqcap \\ & \langle \leq 1 ElementTrigger\_Element \rangle \sqcap \langle \leq 1 ElementTrigger\_Trigger \rangle \end{aligned}$$

$$ElementTriggerRole \equiv ElementTrigger\_Element^{-} \circ ElementTrigger\_Trigger$$

$$\begin{aligned} Class \sqsubseteq & NamedElement \sqcap \neg \langle Property \sqcup Qualifier \sqcup Method \sqcup Trigger \sqcup Schema \rangle \sqcap \\ & \forall PropertyDomain.Property \sqcap \forall MethodDomain.Method \sqcap \\ & \forall Subtype.Class \sqcap \langle \leq 1 Subtype \rangle \sqcap \forall Subtype^{-}.Class \sqcap \\ & \forall Range\_Class^{-}.Range \sqcap \langle \leq 1 Range\_Class^{-} \rangle \sqcap \forall RangeRole.Reference \end{aligned}$$

$$Supertype \equiv Subtype^{-}$$

$$\begin{aligned} Range \sqsubseteq & \exists Range\_Class.Class \sqcap \exists Range\_Reference.Reference \sqcap \langle \leq 1 Range\_Class \rangle \sqcap \\ & \langle \leq 1 Range\_Reference \rangle \end{aligned}$$

$$RangeRole \equiv Range\_Class^{-} \circ Range\_Reference$$

$$Association \sqsubseteq Class \sqcap \forall hasReference.Reference \sqcap \langle \geq 2 hasReference \rangle$$

$$Indication \sqsubseteq Class \sqcap \neg Association$$

$$\begin{aligned} Qualifier \sqsubseteq & NamedElement \sqcap \\ & \exists Value.Variant \sqcap \langle \leq 1 Value \rangle \sqcap \\ & \neg \langle Property \sqcup Class \sqcup Method \sqcup Trigger \sqcup Schema \rangle \sqcap \\ & \exists Characteristics^{-}.NamedElement \sqcap \langle \leq 1 Characteristics^{-} \rangle \end{aligned}$$

$$\begin{aligned} Property \sqsubseteq & NamedElement \sqcap \neg \langle Method \sqcup Trigger \sqcup Schema \rangle \sqcap \\ & \exists PropertyOverride.Property \sqcap \langle \leq 1 PropertyOverride \rangle \sqcap \\ & \forall PropertyOverride^{-}.Property \sqcap \\ & \exists PropertyDomain^{-}.Class \sqcap \langle \leq 1 PropertyDomain^{-} \rangle \end{aligned}$$

$$Method \sqsubseteq NamedElement \sqcap \neg\langle Trigger \sqcup Schema\rangle \sqcap \forall MethodOverride.Method \sqcap \langle \leq 1 MethodOverride\rangle \sqcap \forall MethodOverride^{-}.Method \sqcap \exists MethodDomain^{-}.Class \sqcap \langle \leq 1 MethodDomain^{-}\rangle$$

$$Trigger \sqsubseteq NamedElement \sqcap \neg Schema \sqcap \forall ElementTrigger\_Trigger^{-}.ElementTrigger \sqcap \langle \geq 1 ElementTrigger\_Trigger^{-}\rangle \sqcap \forall ElementTriggerRole^{-}.NamedElement \sqcap \langle \geq 1 ElementTriggerRole^{-}\rangle$$

$$Schema \sqsubseteq NamedElement \sqcap \forall ElementSchema^{-}.NamedElement$$

$$Reference \sqsubseteq Property \sqcap \exists Range\_Class^{-}.Range \sqcap \langle \leq 1 Range\_Reference^{-}\rangle \sqcap \exists RangeRole^{-}.Class \sqcap \langle \leq 1 RangeRole^{-}\rangle \sqcap \exists HasReference^{-}.Association \sqcap \langle \leq 1 HasReference^{-}\rangle$$

A continuación se extiende la expresión $\mathcal{AL}\varepsilon\mathcal{CNOQ}^{-}_{\mathcal{HR}^{+}\circ}$ del meta-esquema CIM con los meta calificadores y los calificadores estándar definidos por el DMTF:

Nuevos conceptos:

$$Aggregation \sqsubseteq Association \sqcap \forall hasReference.Aggregate \sqcap \langle = 2 hasReference\rangle$$

$$Aggregate \sqsubseteq Reference$$

Nuevas restricciones sobre los conceptos existentes:

$$NamedElement \sqsubseteq \forall Description.String \sqcap \langle \leq 1 Description\rangle \sqcap \forall DisplayString.String \sqcap \langle \leq 1 DisplayString\rangle$$

$$Property \sqsubseteq \forall Key.Boolean \sqcap \langle \leq 1 Key\rangle \sqcap \forall MappingStrings.String \sqcap \forall Alias.String \sqcap \langle \leq 1 Alias\rangle \sqcap \forall Counter.Boolean \sqcap \langle \leq 1 Counter\rangle \sqcap \forall Gauge.Boolean \sqcap \langle \leq 1 Gauge\rangle$$

$$Class \sqsubseteq \forall Abstract.Boolean \sqcap \langle \leq 1 Abstract\rangle \sqcap \forall MappingStrings.String$$

3.5. Servicios de razonamiento automático

El "mapping" CIM-$\mathcal{ALECNOQ}^{-}_{\mathcal{H}R^{+}\circ}$ supone una formalización semántica de las conceptualizaciones CIM que permite la realización de servicios de razonamiento automático. La base de conocimiento tiene una semántica que la equipara a un conjunto de axiomas en lógica de predicados de primer orden. Por tanto, del mismo modo que cualquier otro conjunto de axiomas, contiene conocimiento implícito que puede explicitarse a través de la inferencia lógica. El servicio de inferencia fundamental es la verificación de consistencia para bases de conocimiento asertivo (ABox), en función del cual puede expresarse el resto.

3.5.1. Razonamiento acerca de los modelos CIM

El "mapping" de un modelo CIM a una base de conocimiento terminológico (TBox) supone la construcción de una terminología $\mathcal{T}$. Durante la construcción de un modelo CIM es importante descubrir si una nueva clase tiene sentido o, por el contrario, es contradictoria respecto del resto del modelo, en cuyo caso nunca podrá instanciarse de modo consistente. Desde un punto de vista lógico, un nuevo concepto C tiene sentido si existe al menos una interpretación $\mathcal{I}$ que satisface los axiomas de $\mathcal{T}$ y para la que el concepto denota un conjunto no vacío. A esta interpretación se la denomina modelo y se escribe $\mathcal{T} \models C$. A esta propiedad del concepto C con respecto a $\mathcal{T}$ se la denomina *satisficibilidad*.

El diseñador de una conceptualización CIM utilizará los servicios de razonamiento ofrecidos por el subsistema DL de su herramienta de modelado para verificar que todas las clases creadas son satisfacibles respecto del resto del modelo y que se cumplen las relaciones de generalización/especialización esperadas. Todos los servicios de razonamiento estarán basados en los siguientes servicios prototípicos:

Satisficibilidad Un concepto C es satisfacible con respecto a una terminología $\mathcal{T}$ si existe un modelo $\mathcal{I}$ de $\mathcal{T}$ tal que $C^{\mathcal{I}} \neq \varnothing$. Se escribe $\mathcal{T} \models C$. La satisficibilidad de la propia $\mathcal{T}$ se expresa como $\mathcal{T} \models$.

Subsunción Un concepto C es subsumido por un concepto D con respecto a $\mathcal{T}$ si $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ para todo modelo $\mathcal{I}$ de $\mathcal{T}$. Se escribe $\mathcal{T} \models C \sqsubseteq D$. La subsunción se puede expresar en términos de satisficibilidad como $\mathcal{T} \models C \sqsubseteq D \Leftrightarrow \mathcal{T} \not\models C \sqcap \neg D$. Del mismo modo, la satisficibilidad puede expresarse en términos de subsunción como $\mathcal{T} \not\models C \Leftrightarrow \mathcal{T} \models C \sqsubseteq \bot$.

Equivalencia Un concepto C es equivalente a un concepto D con respecto a $\mathcal{T}$ si $C^{\mathcal{I}} = D^{\mathcal{I}}$ para todo modelo $\mathcal{I}$ de $\mathcal{T}$. Se escribe $\mathcal{T} \models CequivD$. La equivalencia

puede expresarse en términos de satisficibilidad como $\mathcal{T} \vDash C \equiv D \Leftrightarrow \mathcal{T} \nvDash C \sqcap \neg D$ y $\mathcal{T} \nvDash \neg C \sqcap D$, y en términos de subsunción como $\mathcal{T} \vDash C \equiv D \Leftrightarrow \mathcal{T} \vDash C \sqsubseteq D$ y $\mathcal{T} \vDash D \sqsubseteq D$.

Disjunción Dos conceptos C y D son disjuntos con respecto a $\mathcal{T}$ si $C^{\mathcal{I}} \cap D^{\mathcal{I}} = \varnothing$ para todo modelo $\mathcal{I}$ de $\mathcal{T}$. La disjunción puede expresarse en términos de satisficibilidad como $\mathcal{T} \nvDash C \sqcap D$, y en términos de subsunción como $\mathcal{T} \vDash C \sqcap D \sqsubseteq \bot$.

3.5.2. Razonamiento acerca del conocimiento asertivo de los agentes de gestión

Los agentes de gestión que siguen el modelo de información *CIMOnt* propuesto manejan tanto una base de conocimiento terminológico (TBox), generalmente obtenida a partir de la representación sintáctica OWL de una conceptualización CIM, como una base de conocimiento asertivo (ABox). Si bien los agentes pueden hacer uso de los servicios de razonamiento comentados en la sección anterior, principalmente el de clasificación de conceptos, es mucho más probable que requieran servicios de inferencia asociados al conocimiento asertivo acerca de los individuos del entorno. El principal servicio de este tipo es el relacionado con la comprobación de la consistencia de la representación de dicho conocimiento desde un punto de vista estrictamente lógico, ya que de otro modo se podrían extraer conclusiones incoherentes. Una vez verificada la consistencia de la base de conocimiento asertivo, un agente podrá inferir conocimiento acerca de las relaciones entre conceptos, roles e individuos (y por tanto clases, asociaciones y objetos CIM) basándose en los siguientes servicios prototípicos:

Consistencia Una base de conocimiento asertivo (ABox) $\mathcal{A}$ es consistente con respecto a una TBox $\mathcal{T}$ si existe al menos una interpretación $\mathcal{I}$ que es a la vez modelo de $\mathcal{T}$ y de $\mathcal{A}$.

Comprobación de instancia Consiste en la comprobación de que una aserción es consecuencia lógica del conocimiento almacenado en la ABox $\mathcal{A}$. Una aserción α es consecuencia lógica de $\mathcal{A}$, y se escribe $\mathcal{A} \vDash \alpha$, si toda interpretación que satisface $\mathcal{A}$ (esto es, todo modelo de $\mathcal{A}$), también satisface α. Si α es $C(a)$, este servicio puede reducirse al servicio de consistencia puesto que $\mathcal{A} \vDash C(a) \Leftrightarrow \mathcal{A} \cup \{\neg C(a)\}$ es inconsistente.

Pertenencia (extensión) Un agente querrá generalmente conocer todos los individuos que son instancia de un determinado concepto (tanto un concepto

básico, como una expresión de concepto). El servicio de pertenencia o extensión permite utilizar el lenguaje de descripción para formular este tipo de consultas. Dada una ABox $\mathcal{A}$, el servicio de extensión recupera el conjunto de individuos $\{a \mid \mathcal{A} \vDash C(a)\}$.

Realización (Clasificación de instancias) Se trata de un servicio de inferencia dual al de extensión: Dado un individuo a y una ABox $\mathcal{A}$, el servicio de realización encuentra el conjunto de conceptos más específicos[4] $\{C \mid \mathcal{A} \vDash C(a)\}$.

[4]En este contexto, se entiende por conceptos más específicos aquellos que son mínimos con respecto al orden inducido por el operador de subsunción $\sqsubseteq$.

Capítulo 4

Modelo de Información: Interfaz de interacción

Índice General

4.1. Del modelado de actos hablados al modelado de conversaciones 97

4.2. Proceso de especificación de una interfaz DIA 99

4.3. Requisitos de un lenguaje de especificación de protocolos 104

4.4. El lenguaje de especificación ACSL 106

4.5. Verificación y evaluación semántica de especificaciones ACSL 121

El modelo de interacción utilizado en el paradigma de agentes se basa en la realización por parte de éstos de *acciones comunicativas* en un intento por influenciar a otros agentes (en su estado y en su comportamiento) de forma apropiada. Este intento por tratar la comunicación como acción tiene sus orígenes en la *teoría de actos hablados* [Aus62, SV85] y se basa en la asunción de que los agentes llevan a cabo las acciones habladas del mismo modo que el resto de acciones; en pro de sus intenciones. Con estas premisas se han creado lenguajes de comunicación de agentes (ACL) que modelan los actos hablados como mensajes tipados que constituyen los bloques de construcción para la comunicación. KQML del KSF [AK93] y FIPA-ACL [FIP02] constituyen los dos principales ejemplos de ACL. Sin embargo, los agentes no participan en intercambios de mensajes aislados, sino que entablan conversaciones, y por tanto, secuencias coherentes de mensajes con el fin de llevar a cabo tareas específicas que requieren coordinación, tales como negociaciones o subastas. Esta secuencia de intercambio puede surgir espontáneamente o puede haber sido acordada a priori y especificada mediante un protocolo de interacción. La especificación a priori de los protocolos de interacción y el consenso acerca del protocolo concreto

a seguir en una determinada conversación facilita el diseño de agentes capaces de entablar conversaciones coherentes con otros agentes.

La especificación de protocolos de interacción no es una idea nueva en el paradigma de agentes. Existen multitud de aproximaciones que han considerado, adoptado y enriquecido sintáctica y semánticamente diferentes técnicas utilizadas tradicionalmente en el diseño de los protocolos de comunicación asociados a los sistemas distribuidos. Algunas de estas aproximaciones han otorgado mayor importancia a la obtención de modelos fáciles de interpretar e implementar; es el caso de las basadas en diagramas de transición de estados [WF86, Har87] y en máquinas de estados finitos [Had96]. Otras, a costa de dificultar su interpretación, han incorporado formalismos matemáticos y/o lógicos más complicados que les han permitido desde expresar concurrencia, hasta validar y simular el protocolo especificado. Entre estas se encuentran las propuestas basadas en Redes de Petri [CCF+99]. Otras propuestas están basadas en lenguajes de especificación sintáctica como LOTOS [Kon99], Estelle o SDL [Tur93]. La mayoría de estas técnicas adolecen de una representación gráfica cercana a la visión que del protocolo tiene el diseñador. Este hecho se acrecienta a medida que aumenta la complejidad de la interacción que se desea modelar. Los Diagramas de Protocolo propuestos en la notación Agent-UML [JOB00] proporcionan una visión mucho más cercana a la del diseñador. Sin embargo, estos diagramas suponen una mera notación adecuada para las fases de análisis y diseño de un método, por lo que carecen de soporte para la especificación formal de los protocolos de interacción en un lenguaje estándar que permita la validación y la simulación de los protocolos, ni tampoco su intercambio, descubrimiento y aprendizaje dinámico por parte de los agentes que deseen conversar. [FIP02] ha contribuido a este esfuerzo de manera global, proporcionando especificaciones en notación Agent-UML de gran número de protocolos interesantes. Cabe considerar también otras técnicas de descripción formal basadas en la lógica como las presentadas en [FW94, Dig00b], si bien están más orientadas a la inferencia automática del patrón de intercambio de mensajes que a su especificación a priori.

Se observa un gran vacío entre las técnicas formales existentes, que implican una elevada complejidad de diseño y las notaciones gráficas que adolecen de semántica precisa, dificultan la demostración automática de propiedades de los protocolos e impiden su interpretación automática por parte de los agentes. Esta tesis se propone reducir este vacío aportando un novedoso lenguaje de especificación formal de interacciones, cuya expresividad sintáctica resulta muy cercana a la visión del protocolo ofrecida por la notación Agent-UML en sus diagramas de interacción y cuya exactitud semántica permitirá automatizar la interpretación de los protocolos

especificados y estudiar algunas propiedades formales de los mismos.

4.1. Del modelado de actos hablados al modelado de conversaciones

En el paradigma de objetos, se denomina interacción al patrón de comunicación seguido por un conjunto de instancias que desempeñan un conjunto de roles para conseguir un propósito específico en el contexto de una colaboración [OMG01a]. Un mensaje define una comunicación entre instancias especificada en el contexto de una interacción. Dicha comunicación suele representar la invocación de una operación (aunque también el lanzamiento de una señal y la creación o destrucción de una instancia) mediante un estímulo, en cuyo caso provoca la realización de la acción (servicio) asociada a dicha operación a través de la ejecución del (cuerpo del) método correspondiente. Este escenario formal se simplifica mediante el concepto "invocación de un método".

El paradigma de agentes atribuye también a éstos la posibilidad de ejecutar acciones internas, que se corresponden vagamente con el concepto de [cuerpo de un] método en terminología de objetos. Sin embargo, en el paradigma de agentes no existe el concepto de "invocación de un método". Esto es debido a que un agente es principalmente un "artifact" software autónomo, y por tanto tiene control total tanto sobre su propio estado como sobre su comportamiento. No se contempla la situación en que un agente ejecuta una acción (servicio) simplemente por el hecho de que otro agente se lo requiere. El locus de control con respecto a la decisión de ejecutar o no una acción es muy diferente en el paradigma de agentes respecto del resto de paradigmas de programación (incluido el de objetos, que es el más próximo en capacidad de abstracción). En general, los agentes no pueden forzar a otros agentes para que realicen determinada acción, ni modificar directamente el estado interno de los mismos. Sin embargo, esto no significa que no puedan comunicarse. Tan solo significa que el modelo de interacción es diferente: en su interacción, los agentes, para alcanzar sus fines, realizan *acciones comunicativas* con otros agentes, en un intento por influenciarles en su estado y en su comportamiento de forma apropiada.

La *teoría de actos hablados* [Aus62, SV85] trata la comunicación como acción. Asume que las acciones habladas se llevan a cabo por los interlocutores del mismo modo que el resto de acciones, en pro de sus intenciones. Este hecho ha provocado que el modelo de interacción del paradigma de agentes se haya basado tradicionalmente en la utilización de lenguajes de comunicación de agentes (ACL) fuertemente influenciados por la teoría de actos hablados. KQML de KSF [AK93] y FIPA-ACL

[FIP02] constituyen los dos principales ejemplos de ACL. En estos lenguajes los actos hablados se modelan como mensajes tipados que constituyen los bloques de construcción para la comunicación.

En una interacción entre dos o más agentes se sucederá un intercambio coherente de mensajes que seguirá un determinado patrón. Un diseñador tiene la opción de hacer emerger espontáneamente dicho patrón de intercambio a partir de las decisiones tomadas por los agentes en su proceso de planificación. Los agentes tomarán estas decisiones en base a su conocimiento acerca de: (1) el significado de los mensajes recibidos, (2) sus propias metas, creencias y otras aptitudes mentales y, posiblemente, (3) las metas, creencias e intenciones de los otros agentes. Sin embargo, esta opción complica sustancialmente, e incluso podríamos decir que limita, el diseño del agente. Una alternativa mucho más pragmática consiste en definir patrones típicos de intercambio y especificarlos de modo que un agente pueda interactuar con otros agentes simplemente siguiendo dicha especificación. El conjunto de interacciones, basadas en el intercambio de actos hablados en forma de mensajes siguiendo un patrón preestablecido, constituye una conversación y a la especificación del patrón de intercambio se la denomina protocolo de interacción.

Una conversación viene definida por tanto por un patrón de intercambio de mensajes que dos o más agentes acuerdan seguir durante su interacción. En este sentido, una conversación es un protocolo de coordinación preestablecido que proporciona contexto al envío y recepción de mensajes, facilitando su interpretación. La especificación a priori de las diferentes conversaciones en las que puede participar un agente a través de protocolos de interacción y el consenso acerca del protocolo concreto a seguir en una determinada interacción facilita el diseño de agentes capaces de entablar conversaciones coherentes con otros agentes.

El conjunto de conversaciones en que puede participar un agente define la *interfaz de comunicación* (y, por tanto, de servicio) con dicho agente. Por tanto, los conjuntos estandarizados de protocolos de interacción que especifican dichas conversaciones pueden verse como *Definiciones de Interfaz de Agentes* (DIAs), del mismo modo que los conjuntos de definiciones de procedimientos y funciones suponen las interfaces de definición y/o bibliotecas de programación en otros paradigmas de programación. La interfaz de un rol del agente (y por tanto un tipo de comportamiento) puede especificarse mediante la identificación de un subconjunto de protocolos adecuados para influenciar dicho comportamiento. Cualquier agente que presente este rol tan sólo precisará implementar el conjunto apropiado de conversaciones y ofrecerlo como su interfaz de comunicación. Por todo ello, se precisa de lenguajes de especificación formal que permitan definir con claridad y precisión estas interfaces (LDIA) de

modo que puedan ser utilizadas, tanto por los diseñadores y programadores en su fase de desarrollo, como por los propios agentes, de forma automática, durante sus interacción.

4.2. Proceso de especificación de una interfaz DIA

La siguientes secciones presentan un lenguaje de especificación formal de conversaciones entre agentes denominado ACSL (del inglés *Agent Conversation Specification Language*). ACSL define una sintaxis abstracta que establece un vocabulario apropiado para describir, de forma estándar y formal, los aspectos contractuales de los protocolos de interacción que especifican dichas conversaciones. La especificación sintáctica en ACSL de un protocolo de interacción constituye parte del proceso de especificación de una interfaz DIA y, por tanto, parte del lenguaje LDIA.

Con el fin de ilustrar con un ejemplo práctico la descripción formal del lenguaje ACSL y el proceso de especificación de una interfaz DIA mediante dicho lenguaje, se describe a continuación uno de los procesos de negociación que se han utilizado en el diseño de las holarquías de gestión que se introducen en el capítulo 5 para distribuir las [sub]tareas a realizar de forma cooperativa por los diferentes agentes (atómicos u holones) de gestión que conforman una organización holónica. El proceso de distribución de tareas consta de dos fases caracterizadas por sendos protocolos de interacción y, por tanto, por sendas interfaces LDIA que deberán ser implementadas por los agentes participantes.

En la *fase de asignación*, el grupo de agentes, coordinado por el agente de interfaz o *cabeza* del holón, utiliza una versión modificada del protocolo *IteratedContractNet* propuesto por FIPA para realizar de forma cooperativa la distribución de las tareas. Tras esta fase de asignación, alguno de los agentes puede fallar en su intento por alcanzar la meta que le fue encomendada (bien porque no reservó recursos al hacer su propuesta y ahora no dispone de los recursos necesarios, bien porque el coste de realización no es ahora el mismo que cuando se hizo la propuesta) iniciándose entonces una *fase de compensación* bipartita con el fin de permitir al agente *cabeza* y al agente implicado renegociar una nuevas condiciones para la realización de la tarea. Entre las condiciones negociadas se incluyen nuevos plazos de finalización, recursos asignados, costes implicados, etc. Si esta fase finaliza sin alcanzarse un acuerdo, se inicia una nueva fase de asignación para dicha tarea.

En la fase de asignación, el agente que actúa como cabeza del holón difunde una solicitud de propuestas a todos los miembros del holón mediante un acto hablado *cfp*. Estos últimos evaluarán la solicitud y enviarán sus propuestas (acto *propose*) en caso de que que estén dispuestos a realizar la tarea solicitada en unas condiciones

determinadas. En otro caso rechazarán la solicitud (acto *refuse*). El agente *cabeza* evaluará entonces las propuestas recibidas y decidirá aceptar una (o más) de éstas (*accept*) y rechazar las restantes (*refuse*), o bien iniciar una nueva fase de solicitud modificando la solicitud inicial en busca de mejores propuestas. En este último caso se reduce el número de participantes, por lo que no puede hablarse ya de un protocolo de difusión y sí de un protocolo multipunto. El protocolo puede finalizar también si en algún momento se rechazan todas las propuestas.

La fase de compensación comienza cuando un agente, tras fallar en la realización de la tarea asignada y notificárselo al agente *cabeza*, o bien tras el vencimiento de un temporizador en este último sin recibir dicha notificación, recibe del agente *cabeza* una propuesta para realizar dicha tarea bajo unas ciertas condiciones (a diferencia de la fase de asignación, que se iniciaba con la difusión de una solicitud de propuestas, ahora el agente de interfaz del holón construye la propuesta inicial o propuesta de compensación). El agente implicado debe entonces evaluar la propuesta y tomar una decisión, pudiendo como consecuencia aceptar la tarea, rechazarla (en cuyo caso finaliza el protocolo y, posiblemente, se inicia una nueva fase de asignación para dicha tarea) o realizar una contra-propuesta (*counter-propose*) que satisfaga parcialmente la propuesta original. En caso de aceptar la propuesta, el agente tiene aún la libertad de desdecirse y cancelar su aceptación (nivel de deseos) mediante un acto *cancel*, o bien comprometerse a satisfacer la propuesta (nivel de intenciones) mediante un acto *commit*. Si el agente acuerda satisfacer la propuesta, se habrá producido una [nueva] asignación correcta. El agente deberá entonces informar acerca de la terminación de dicha asignación, comunicando si esta ha sido satisfecha (*inform* o *satisfy*) o ha fallado (*Failure*). También se contempla la posibilidad de que un agente cancele un compromiso contraído (*de-commit*). Esta cancelación conlleva diferentes consecuencias que la cancelación de una aceptación (*cancel*). Si se realiza una contra-propuesta, el agente iniciador deberá opinar acerca de la misma, bien aceptándola, bien rechazándola, en cuyo caso, tiene la opción de realizar una nueva contra-propuesta o terminar la fase de compensación y, posiblemente, volver a la fase de asignación.

La utilización de esta fase de compensación evita la reserva ineficiente de recursos en la fase de asignación (múltiples agentes reservan recursos al hacer sus propuestas y sin embargo sólo se asigna la tarea a uno de ellos, situación que empeora si se tiene en cuenta que un agente puede participar simultáneamente en más de un proceso de asignación o incluso la posible aplicación en cascada del protocolo de asignación en una organización holónica recursiva) y permite la asignación subóptima (un agente recibe múltiples solicitudes de propuestas, pero no puede atender todas las aceptadas

con el coste propuesto, por lo que desearía renegociar el coste de éstas últimas mientras éste sea inferior al propuesto inicialmente por los demás agentes en la fase de asignación).

Esta descripción del proceso de negociación de tareas, asume la existencia de un conjunto de actos hablados, de mayor nivel que los disponibles en KQML o FIPA-ACL, que definen [un subconjunto de] las acciones comunicativas disponibles en cada uno de los agentes que ofrece este protocolo como interfaz de comunicación. A continuación se describe de manera informal la semántica de cada uno de estos actos hablados (representados como performativas del ACL correspondiente):

Propose: Se utiliza para proponer a un agente una submeta a conseguir.

Counter-Propose: Una contra-propuesta consiste en la propuesta de una nueva submeta que satisface parcialmente la meta de una propuesta inicial. La utilización de este acto hablado puede conllevar un ciclo de negociación con sucesivas contra-propuestas por ambas partes.

Accept: Se utiliza para notificar la aceptación de una [contra]propuesta.

Reject: Se utiliza para notificar el rechazo de una [contra]propuesta. Conlleva el comienzo de una nueva fase de negociación.

Cancel: Cancela una [contra]propuesta previamente aceptada.

Commit: Confirmación de que el agente acuerda (tiene la intención de) satisfacer una propuesta. Finaliza una fase de [re]negociación.

De-Commit: Cancela un Commit previo.

Satisfy: Anuncia que se ha conseguido una meta solicitada previamente.

Failure: Informa de que ha fallado la ejecución de una meta acordada previamente.

A la vista de la descripción en lenguaje natural presentada sobre estas líneas, puede concluirse que, pese a que ésta proporciona una idea más o menos precisa de cómo es el proceso de coordinación, en todo caso deja bastante margen para imprecisiones y ambigüedades que originarán numerosos problemas a la hora de diseñar e implementar agentes capaces de interactuar y coordinarse en base a este protocolo. Más aún si el diseño de cada uno de los agentes participantes se realiza de forma independiente por distintos equipos de trabajo, situación típica en entornos abiertos y distribuidos de Internet.

La figura 4.1 muestra una especificación más formal del protocolo de interacción descrito en la fase de compensación mediante una máquina de estados finitos, que detalla el intercambio de mensajes desde el punto de vista del agente que recibe la propuesta inicial.

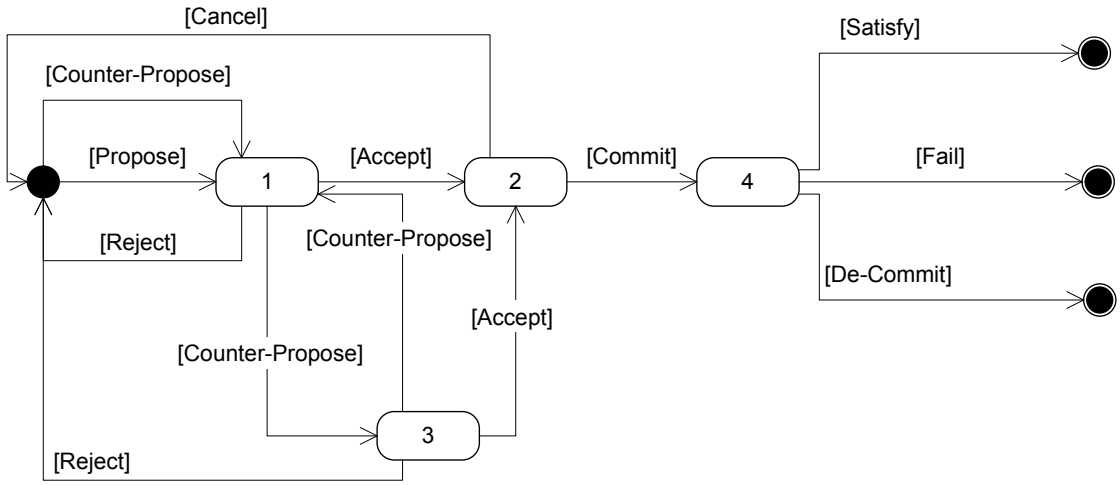

Figura 4.1: Especificación FSM del protocolo utilizado para negociar la asignación de tareas dentro de un holón de gestión [Fase de compensación]

Si bien la utilización de máquinas de estados finitos supone un mayor grado de formalización, dista mucho aún de ser completa. Por una parte, la máquina de estados finitos mostrada en la figura 4.1 describe el intercambio de mensajes tan solo desde el punto de vista del agente que recibe la propuesta de compensación inicial. La máquina de estados finitos correspondiente al agente que inicia el protocolo difiere de ésta, lo cual introduce nuevos problemas a la hora de verificar/validar las especificaciones. Por otra parte, no se distingue entre mensajes enviados y recibidos, ni se dice nada acerca de qué propuesta está siendo negociada (téngase en cuenta que la vuelta a un estado anterior tras un *CounterPropose* implica un cambio en dicha propuesta. Por último no existe soporte para concurrencia y sincronización, necesaria en conversaciones multi-agente.

Las redes de Petri constituyen otra forma de especificación formal más compacta que la utilizada en autómatas finitos y, adicionalmente, permite expresar el sincronismo necesario entre los autómatas que ejecutan en paralelo. Más aún, facilita las labores de validación gracias al fuerte formalismo matemático subyacente en ella. Sin embargo, su utilización en el modelado de protocolos de interacción es prohibitiva debido a la complejidad de su representación.

La especificación AUML mostrada en las figuras 4.2 y 4.3 proporciona una vi-

sión mucho más clara del proceso de interacción que deben seguir los agentes [1]. AUML, pese a que permite representar protocolos anidados como subprotocolos, no proporciona soporte para la representación gráfica de excepciones y protocolos de compensación (éstos, a diferencia de aquellos, no ocurren en un momento preciso, sino de manera asíncrona). Es por ello que no se observa nexo entre las dos figuras. Además, presenta numerosas carencias y lagunas, tal y como se ha comentado anteriormente. De una parte, se trata de una mera notación gráfica, por lo que los agentes no pueden utilizarla en tiempo de ejecución para comunicarse, descubrir y "aprender" nuevos protocolos de interacción. De otra parte, adolece de una semántica precisa que permita su simulación o el estudio de algunas propiedades tales como su terminación en tiempo finito, dead-locks, etc.

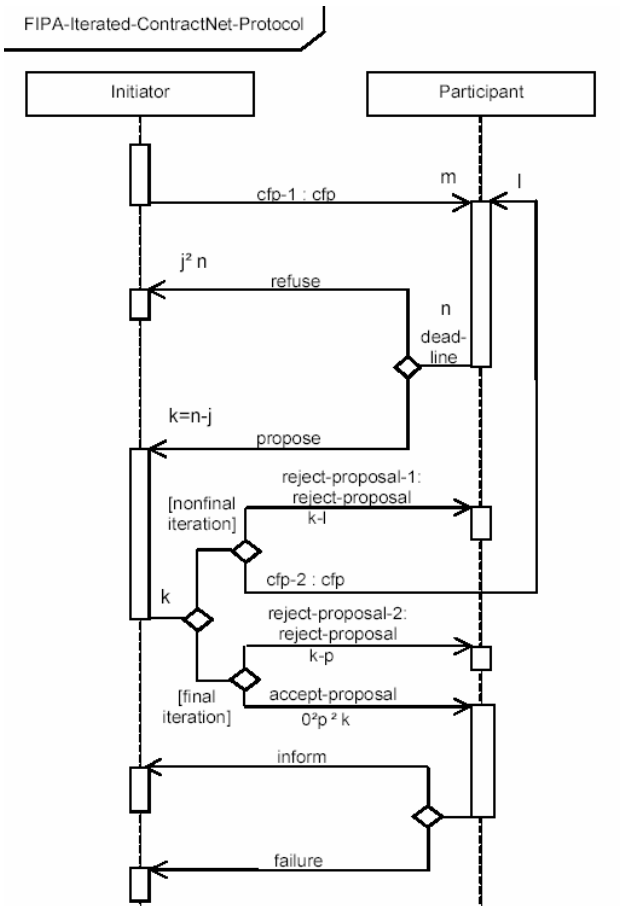

Figura 4.2: Especificación AUML del protocolo para negociar la asignación de tareas entre los miembros de un holón de gestión (Fase de asignación)

La siguiente sección introduce los requisitos necesarios de un lenguaje de especi-

[1]La figura 4.3 ha sido capturada de la herramienta $ACSL^{+}$, desarrollada por el autor como parte de un marco de trabajo destinado a la validación de la arquitectura propuesta

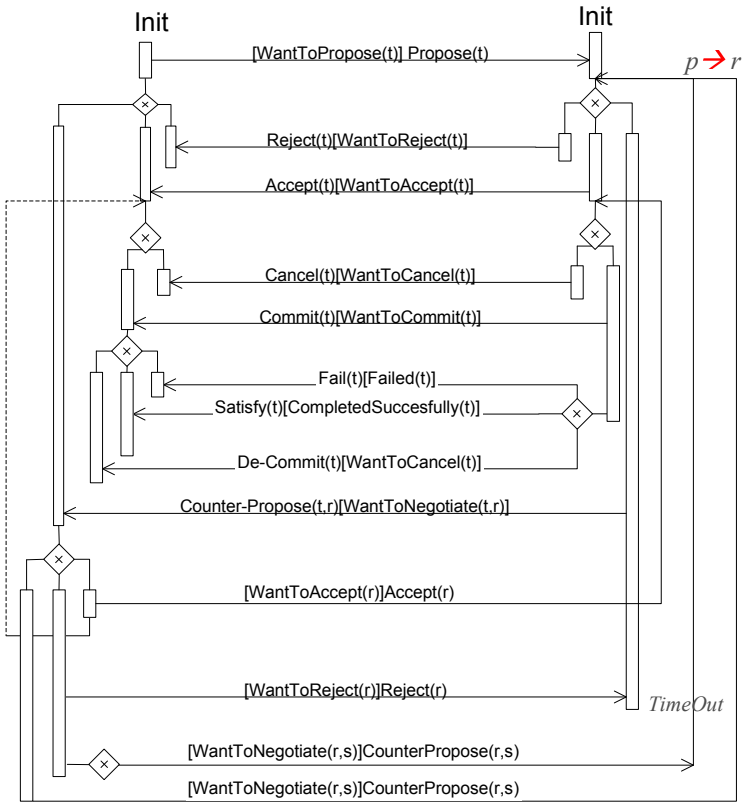

Figura 4.3: Especificación AUML del protocolo para negociar la asignación de tareas entre los miembros de un holón de gestión (Fase de compensación)

ficación que se proponga soslayar estas carencias, como es el caso del lenguaje que se propone en la sección 4.4.

4.3. Requisitos de un lenguaje de especificación de protocolos

La especificación formal de un protocolo de interacción es un proceso complejo. El lenguaje de especificación debe cubrir todos los aspectos de una conversación real, para lo cual debe ofrecer construcciones sintácticas que permitan expresar:

- Los roles que participan en la interacción, cuyas caracterizaciones y asociaciones definen el marco de colaboración en que ocurre dicha interacción. La caracterización de estos roles y sus asociaciones suponen una proyección del modelo estático del sistema sobre la colaboración en cuestión.

- las diferentes secuencias de intercambio entre las que puede optar un agente a lo largo de la interacción y/o en las que puede participar simultáneamente,

así como su sincronización,

- los intercambios de mensajes que se producen en cada una de estas secuencias,
- las condiciones bajo las cuales se producen estos intercambios, y por tanto el envío concurrente de múltiples mensajes o el envío selectivo en base a un proceso de decisión,
- las características de los mensajes intercambiados: sus multiplicidades (*unicast*, *multicast*, *broadcast*), tipo de envío (síncrono, asíncrono o retardado) y tipo de concurrencia,
- las temporizaciones que deben contemplarse y asegurarse (*deadlines* y *timeouts*),
- la composición de protocolos a partir de otros protocolos de modo que se puede especificar el patrón de interacción de forma modular y
- la parametrización de protocolos con el fin de facilitar su reutilización.

Además de estas propiedades básicas comúnmente reconocidas por la comunidad internacional, a juicio del autor deben considerarse otro conjunto de propiedades con el fin de poder especificar otros aspectos de una interacción compleja:

- Generalmente, los agentes entablan simultáneamente múltiples diálogos con otros agentes. Se hace necesario por tanto especificar de manera clara y flexible la correlación y causalidad de los mensajes, necesaria para ubicar un mensaje en el flujo de la interacción, para asociar cada mensaje a la instancia de PI concreta a la que pertenece, para relacionar mensajes de distintos protocolos de interacción que componen una interacción.
- La especificación debe recoger toda la información relacionada con la semántica del protocolo (proposición acerca de la cual se está opinando, tarea que está siendo negociada, etc.).
- Además de permitir la descripción del patrón básico de intercambio de mensajes, se debe contemplar también la especificación de excepciones de protocolo en dicho patrón y su correspondiente manejo: la recuperación ante la llegada de mensajes fuera de secuencia como consecuencia del carácter autónomo de los interlocutores, la pérdida de mensajes, timeouts, llegada de determinados mensajes, etc.

- También debe permitir expresar contextos transaccionales en el patrón principal y asociar a éstos protocolos de compensación (cancelaciones, renegociaciones, etc.).

4.4. El lenguaje de especificación ACSL

ACSL define una sintaxis abstracta (recogida en el apéndice C) que establece un vocabulario que permite describir de forma estándar y formal los aspectos públicos/contractuales de los protocolos de interacción (PI), de modo que pueda utilizarse por bibliotecas y herramientas de diseño, implementación y monitorización de su ejecución. El protocolo se centra en la especificación del comportamiento comunicativo que dos o más agentes deben respetar si desean interactuar (coordinarse, negociar, etc.) con coherencia. ACSL separa la implementación de los PI interna a los agentes de su descripción externa. Este aspecto es crucial a la hora de facilitar la interoperabilidad en la comunicación entre grupos de agentes heterogéneos y/o que ejecutan en agencias (plataformas) heterogéneas. Se centra en los mensajes ACL para especificar el flujo de mensajes que representa un PI entre dos o más agentes, sin requerir ningún mecanismo de implementación específico.

ACSL define el comportamiento comunicativo de los agentes participantes en un protocolo de interacción sin referenciar ningún tipo de información intercambiada y/o de estado (interno), salvo aquella intrínsecamente relacionada con la semántica del protocolo, a pesar de que es esta información la que conduce en gran medida dicho comportamiento. Por ejemplo, en un PI de negociación para la provisión de un servicio el agente que recibe la solicitud del servicio responderá bien con un mensaje de aceptación, bien con un mensaje de rechazo o desconocimiento en base a un conjunto de criterios. Obviamente, los procesos de decisión seguidos por este último agente deben ser opacos a la especificación del PI, pero el hecho de la existencia de una decisión y la información asociada a dicha decisión relacionada con la semántica del protocolo (el servicio en cuestión) deben reflejarse como alternativas de comportamiento en el PI con una condición opaca. En resumen, el foco del protocolo recae en el comportamiento visible en forma de secuencias de mensajes intercambiados y no en cómo se configuran dichas secuencias a partir de aspectos internos a los agentes participantes.

En ACSL se especifica de forma independiente el comportamiento de cada agente autónomo participante en una conversación y se asume que la única interacción entre estos últimos ocurre a través de intercambios de mensajes. La descripción completa de una conversación mediante un protocolo de interacción muestra no solo el comportamiento de cada participante, sino también cómo ajustan estos comporta-

mientos para producir el comportamiento comunicativo global expresado por dicho protocolo de interacción.

4.4.1. La conversación como composición de protocolos de interacción

El apéndice C recoge la especificación en ACSL correspondiente al rol iniciador de la figura 4.2. La figura 4.4 resume en notación textual dicha especificación, resaltando los distintos bloques de que consta una especificación ACSL.

```
(protocol
  :name fipa-contract-net
  :header ()
  :orchestation
    (context
      (thread-of-interaction
        :thread-name activation
        (exchange
          :message cfp
          :direction in
          :mode activation)
        (switch
          :multichoice false
          :body (branch
                  :condition cfp-eval-not-accepted
                  :action (thread-of-interaction
                    :thread-name refuse
                    :body (exchange
                            :message refuse
                            :direction out
                            :mode terminal))
                (branch
                  :condition cfp-eval-accepted
                  :action (thread-of-interaction
                    :thread-name proposal
                    :body (exchange
                            :message propose
                            :direction out
                            :mode middle)
                (branch
                  :condition default
                  :action (thread-of-interaction
                    :thread-name not-understand
                    :body (exchange
                            :message not-understood
                            :direction out
                            :mode terminal))))
      (pick
        :body (eventHandler
                :event (exchange
                         :message accept-proposal
                         :direction in
                         :mode middle)
        :action (switch
                  :multichoice false
                  :body (branch
                          :condition cnet-done
                          :action (thread-of-interaction
                            :thread-name informing
                            :body (exchange
                                    :message inform
                                    :direction out
                                    :mode terminal))
                        (branch
                          :condition default
                          :action (thread-of-interaction
                            :thread-name failure
                            :body (exchange
                                    :message failure
                                    :direction out
                                    :mode terminal)))
          (eventHandler
            :event (exchange
                     :message reject-proposal
                     :direction in
                     :mode terminal)
            :action (empty))
        ))
        :exception (pick
                     :body (eventHandler
                             :event (exchange
                                      :message cancel
                                      :direction in
                                      :mode terminal)
                             :action (thread-of-interaction
                               :thread-name cancelling
                               :action (protocol-ref
                                         :id tns:cancel-meta-protocol)
                                 (compensate
                                   :id recover))))
        :compensation (transaction
                        :transaction-name recover
                        :body (protocol-ref
                                :id tns:compensation-protocol)))))

(protocol
  :name cancel-meta-protocol
  :header ()
  :orchestation
    (context
      (thread-of-interaction
        :thread-name activation
        (switch
          :multichoice false
          :body (branch
                  :condition cnet-finalized
                  :action (thread-of-interaction
                    :thread-name inform-term
                    :body (exchange
                            :message inform
                            :direction out
                            :mode terminal)))
                (branch
                  :condition default
                  :action (thread-of-interaction
                    :thread-name inform-fail
                    :body (exchange
                            :message failure
                            :direction out
                            :mode terminal)))
      ))
      :exception ()
      :compensation ()))

(protocol
  :name compensation-protocol
  :header ()
  :orchestation
    (context
      (thread-of-interaction
        :thread-name activation
        :body (exchange
                :message inform
                :direction out
                :mode terminal))
      :exception ()
      :compensation ()))
```

Figura 4.4: Representación textual de la especificación ACSL del protocolo Contract-Net

En esta figura puede apreciarse como la especificación consta de un nombre (*name*), una cabecera (*header*) y un cuerpo (*orchestation*) definidos en el contexto de un elemento de bloque *protocol*. El elemento *name* permite referenciar el protocolo desde otros que deseen anidarlo o entrelazarlo. En el elemento *header* aparecen declarados los conjuntos de correlación (*correlationSetDecl*) y las propiedades (*messagePropertyRef*) utilizadas en los intercambios de mensajes para su correlación y enlace dinámico y para la especificación de elementos semánticos, respectivamente, tal y como se explicará más adelante. El cuerpo del protocolo contiene la especificación del patrón básico de intercambio (bloque *orchestation*). El cuerpo del protocolo está constituido por la composición de múltiples elementos *threadOfInteraction* que

se bifurcan y reagrupan para describir el comportamiento comunicativo del agente. El elemento *threadOfInteraction* permite especificar directamente un patrón de intercambio o referenciar mediante un nombre calificado una especificación *protocol* definida en otra especificación diferente. El diagrama de clases UML presentado en la figura 4.5 describe la sintaxis abstracta del lenguaje de especificación ACSL de la que deriva la notación XML empleada en el apéndice A. Los siguientes subapartados describen cada una de las construcciones del lenguaje que permiten especificar un patrón de intercambio.

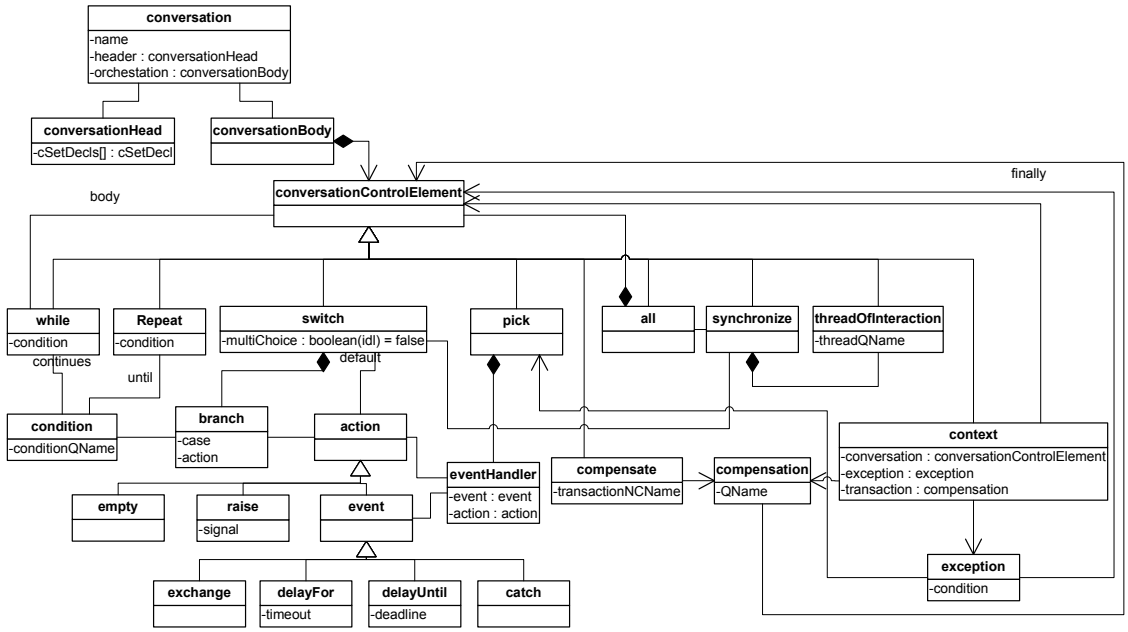

Figura 4.5: Elementos de la sintaxis abstracta del lenguaje ACSL

4.4.1.1. Acciones atómicas

Las acciones atómicas son los elementos básicos a partir de los cuales se construye la especificación de un patrón de intercambio. ACSL contempla tres clases de acciones básicas tal y como muestra la jerarquía de herencia de la clase action en la figura 4.5: los intercambios de mensajes (*exchange*), el lanzamiento de excepciones de protocolo (*raise*) y el vencimiento de temporizadores (*delayFor*, *delayUntil*). La figura muestra también la relación de estas acciones con el resto de construcciones del lenguaje que se explicarán más adelante.

4.4.1.2. Intercambio de mensajes

Los intercambios de mensajes (elemento exchange) son las acciones atómicas fundamentales en una interacción de agentes. ACSL tan solo incluye las propiedades del intercambio que forman parte de la especificación del protocolo. La figura 4.6 muestra un diagrama UML con los elementos del lenguaje relacionados con la especificación de un intercambio de mensaje.

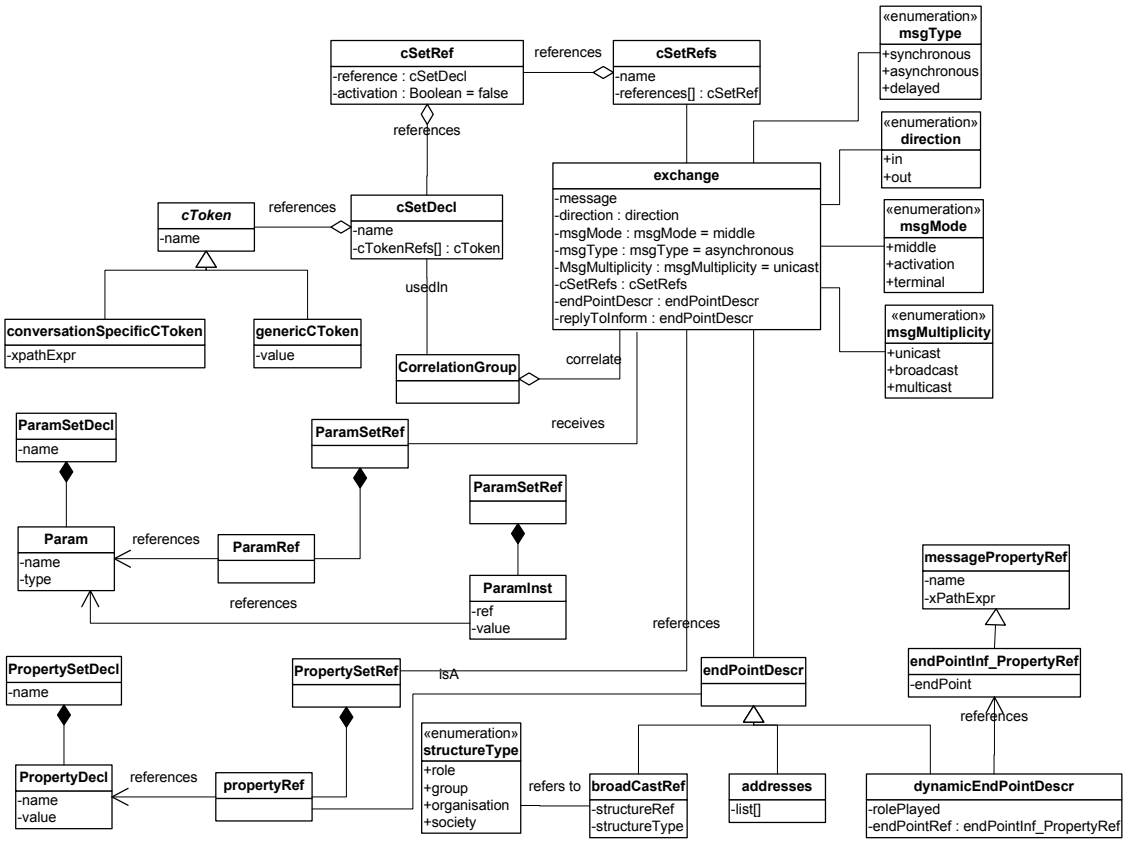

Figura 4.6: Elementos ACSL para la descripción de los intercambios de mensajes correlacionados

El elemento *message* referencia el speech act comunicado. El elemento *direction* indica si se trata de una recepción o de un envío. El modo de intercambio (*msgMode*) especifica si se trata de un mensaje de inicio de protocolo, intermedio o de finalización de protocolo. Un protocolo se instancia de forma implícita con la ocurrencia de uno de los intercambios denotados en la definición XIDL como activation. La instancia termina cuando ocurre uno de los intercambios marcados en la definición XIDL como terminal. El tipo de intercambio (*msgType*) indica si el envío debe ser síncrono, asíncrono o retardado. Por último, la multiplicidad (*msgMultiplicity*) indica si se

trata de un intercambio unicast, multicast o broadcast. Los atributos de envío tipo y multiplicidad sólo tienen sentido en intercambios de salida (envíos).

El elemento *endPointDescr* permite especificar el/los destinatarios de los mensajes en los intercambios de salida. El contenido de este elemento está estrechamente relacionado con la multiplicidad del envío. Si el mensaje es de tipo broadcast, el elemento *broadcastRef* especificará su ámbito (*role*, *group*, *organisation*, *society*). Si se trata de un envío *unicast* o *multicast*, el elemento *addresses* especificará los destinatarios del envío. En el caso de protocolos que requieren configuración dinámica de participantes, se hará uso del elemento *dynamicEndPointDescr*. Este elemento especifica el rol que deben desempeñar los destinatarios del mensaje y referencia un conjunto de propiedades *endPointInfPropertyRef* que permite determinar en qué parte de un intercambio anterior se pueden obtener los destinatarios del mensaje en cuestión.

Como se verá más adelante, un intercambio está relacionado con uno o más elementos que permiten ubicarlo en el contexto del protocolo al que pertenece y en el contexto del resto de protocolos que configuran la interacción global.

4.4.1.3. Lanzamiento de excepciones

La acción básica *raise* de la figura 4.5 denota el lanzamiento de una excepción relacionada con el protocolo de interacción y permite modelar situaciones reales que requieran acciones de cancelación, terminación anormal o inesperada de dicho protocolo. La acción ocasiona la invocación del protocolo de interacción que permite a los agentes recuperarse de situaciones anómalas o finalizar si no es posible la recuperación. Este protocolo se especifica mediante un elemento de bloque *exception* tal y como se explica más adelante.

4.4.1.4. Especificación de temporizaciones

Se proporcionan dos acciones básicas que permiten incorporar el elemento tiempo en la especificación de un protocolo de interacción. La acción *delayFor* permite especificar una pausa por un tiempo determinado y finito, mientras que la acción *delayUntil* permiten establecer deadlines.

4.4.2. Construcciones del lenguaje para la especificación de comportamiento

Los *threadOfInteraction* combinan acciones básicas y otros *threadOfInteraction* mediante un conjunto de construcciones para configurar el protocolo de interacción. Esta sección describe las construcciones básicas que pueden formar parte de

un threadOfInteraction. Las construcciones *context*, *compensation*, *compensate* y *exception* se describen en las siguientes secciones.

ThreadOfInteraction: Un *threadOfInteraction* contiene cero o más acciones, referencias a subprotocolos y construcciones básicas que son ejecutadas secuencialmente y en el mismo orden en que aparecen referenciadas. La secuencia finaliza cuando lo hace el último subelemento.

Empty: El hilo de ejecución más simple es aquel que no contiene ninguna acción y termina tan pronto como comienza. Juega el mismo papel que las sentencias null en los lenguajes de programación.

Switch: Un *switch* permite expresar un comportamiento condicional equivalente al del conector XOR de la notación Agent-UML. Las ramas del *switch* se consideran en el orden en que aparecen. La primera rama cuya condición se haga verdadera proporciona el comportamiento resultante del switch. Si no se toma ninguna rama con regla, entonces se tomará la rama por omisión. Si no existe rama por omisión es que a ésta se le presupone un comportamiento *empty*. El comportamiento del *switch* se completa cuando se completa el comportamiento de la rama seleccionada. El atributo booleano *multiChoice* permite al *switch* expresar también un comportamiento condicional equivalente al del conector OR de la notación Agent-UML.

Pick: La construcción *pick* permite expresar esperas por precondiciones. Se espera la llegada de un suceso (o conjunto de sucesos) y entonces se ejecuta un comportamiento asociado a dicho suceso. Los posibles sucesos son la llegada de algún mensaje o el final de una acción *delay*. El comportamiento finaliza cuando una de las ramas ha sido disparada por la ocurrencia del suceso asociado y el comportamiento asociado a dicha rama ha finalizado. Si se especifica el modificador **times**, el *pick* se repetirá un número predefinido de veces (por ejemplo, para especificar la espera por n mensajes) antes de finalizar ejecutando el comportamiento asociado a la construcción **onTimes**.

All: La construcción *all* permite ejecutar de forma concurrente un conjunto de comportamientos. No impone ningún orden temporal en dicha ejecución. *All* expresa la semántica del conector AND de la notación Agente-UML.

While: La construcción *while* permite repetir un número, indefinido a priori, de veces el patrón de intercambio determinado por un hilo de interacción. El

comportamiento especificado se ejecuta hasta que la condición (*condition*) dada deja de ser verdadera. La condición es opaca, tal y como se ha comentado anteriormente.

Repeat: La construcción *repeat* permite repetir un número preestablecido de veces un patrón de intercambio determinado por un hilo de interacción. El número de veces que se repite es opaco.

Synchronize: La construcción *Synchronize* permite establecer el conjunto de hilos de interacción que deben ser sincronizados tras un *All*, un *Switch* multiopción o un *Or*.

4.4.3. Manejo de excepciones en el patrón de secuenciamiento

En cualquier punto de un protocolo de interacción puede ocurrir la llegada de un mensaje fuera de secuencia, se puede alcanzar un timeout o, incluso, puede ocurrir un error interno que tenga una incidencia directa en el patrón de intercambio, por lo que pueda preverse su tratamiento a nivel de protocolo. Puesto que estos casos no están directamente ligados al flujo normal de la interacción, se denominarán *excepciones de protocolo*, para diferenciarlas de las excepciones que puedan ocurrir en el estado y/o comportamiento autónomo de los agentes y cuya gestión forme parte de los procesos internos de dichos agentes y no esté estrechamente relacionada con la especificación del protocolo de interacción. Con el fin de flexibilizar el manejo de las excepciones de protocolo, éstas se declaran a nivel de contexto (bloque *context*), de modo que su alcance se restringe al subconjunto del patrón de intercambio especificado en dicho contexto.

El manejo de una excepción se asocia a un elemento de bloque *exception* declarado tras la definición del patrón de flujo (bloque *orchestation*) del *context* al que se desea asociar la excepción. El bloque exception contendrá un control de proceso pick cuyas alternativas referencian los diferentes sucesos que originan el lanzamiento de las diferentes excepciones de protocolo que pueden ocurrir durante el flujo definido en la orquestación del contexto y que se desea poder manejar. El proceso *pick* declarado en un bloque exception de un contexto estará disponible desde el momento en que comience el flujo definido en el bloque *orchestation* de dicho contexto. Si ocurre uno de los sucesos declarados en una de las alternativas de dicho *pick*, se parará el flujo normal del contexto y se pasará el control al proceso definido en el cuerpo de la alternativa. Si el flujo normal del contexto termina sin la ocurrencia de sucesos excepcionales, el proceso pick del bloque exception finalizará sin ejecutar

ninguna de sus alternativas. Un elemento típico en la definición de las alternativas de un bloque de excepciones es *compensate*. Este elemento, como se verá en el siguiente apartado permite referenciar (y por tanto invocar) bloques de compensación *compensation* de los contextos anidados, en orden inverso a su terminación. En el apéndice C se recoge la especificación ACSL del protocolo IteratedContractNet para el agente coordinador. Esta especificación muestra la invocación de un protocolo de excepción desde el bloque *exception* del patrón básico.

4.4.4. Protocolos de compensación

Ante la ocurrencia de un error durante una interacción compleja que implica uno o más protocolos, el agente implicado debería tener la oportunidad de iniciar una conversación caracterizada por uno o más protocolos de interacción apropiados que compensasen dicho error en la medida de lo posible, llevando al sistema a un estado próximo al que tenía al comienzo de la interacción. A esta conversación se la denomina *protocolo de compensación*. Este planteamiento no se corresponde con el concepto de transacción ACID tal y como se entiende en el contexto de los sistemas operativos y las bases de datos, ya que una interacción entre agentes autónomos presenta dos características singulares que dificulta el manejo de estas últimas: (1) las interacciones pueden alargarse notablemente en el tiempo haciendo inviable el manejo de los recursos necesarios (cerrojos, copias, etc.) para asegurar el estado de una transacción global a nivel de interacción y (2) la autonomía intrínseca a los agentes impide que pueda asegurarse el "rollback" de la transacción desde una perspectiva centralizada, a pesar de que estos presenten un comportamiento racional.

Para entender mejor el concepto, suponga una conversación entre un agente cliente que pretende negociar la contratación de un servicio que implica a su vez la contratación a través de la figura de un agente intermediario de servicios ofrecidos por otros agentes. Si la negociación entre el agente intermediario y los agentes que proveen los servicios se efectúa en varios pasos sucesivos y ocurre un error en alguno de los pasos intermedios, el agente intermediario se verá abocado a anular las operaciones consensuadas (commited) en los pasos anteriores, para lo cual desearía disponer de un conjunto de protocolos de interacción, previamente consensuados con cada uno de los agentes afectados, que le permitiese llegar, si no al estado inicial, sí a un estado de compromiso. Para ello, por ejemplo, puede iniciar protocolos de cancelación como los presentados por [BF95].

Las compensaciones representan un subconjunto de las excepciones de protocolo, ya que está caracterizado por la gestión de determinado tipo de errores tanto en el seguimiento normal del protocolo como en el estado y/o comportamiento de los

agentes. Sin embargo, conviene diferenciar ambos conceptos. Generalmente, una compensación suele tener un ámbito mayor: suele afectar a múltiples protocolos en el contexto de una interacción, algunos de ellos incluso finalizados, y suele implicar un mayor número de participantes. Además, suele importar el orden en que se lleven a cabo los protocolos compensadores, ya que se suele requerir su ejecución en el orden inverso en que se completaron.

El comportamiento de compensación es en sí mismo parte del protocolo de interacción y debe por tanto especificarse explícitamente. Para ello se utilizan los contextos, que determinan la parte del flujo de interacción que puede requerir de un determinado protocolo compensación (y por tanto su alcance). Un protocolo de compensación se define en un contexto mediante un bloque *compensation* con nombre. La invocación de un bloque *compensation* se realiza de manera explícita mediante el proceso *compensate*, que referencia el nombre de dicho bloque. Un protocolo de compensación solo podrá ser invocado desde un bloque *exception* en su mismo nivel (y por tanto mismo contexto), desde un bloque *exception* en un nivel superior de anidamiento y desde un bloque de compensación en un nivel superior de anidamiento.

Los mensajes enviados como consecuencia de la ejecución de un protocolo de compensación suelen identificar una alternativa de un bloque de excepciones en el protocolo seguido por el agente receptor, permitiendo así a éste iniciar la ejecución de la parte del protocolo de compensación global que le corresponde. En el apéndice C se recoge la especificación ACSL del protocolo IteratedContractNet para el agente que asume el rol coordinador. En esta especificación se muestra la invocación (*compensate*) de un protocolo de compensación *TaskCompensation* referenciado en el bloque *compensation*. La especificación de este protocolo para el rol iniciador, que es el que se corresponde con el rol *coordinador* del protocolo IteratedContractNet aparece también más adelante en dicho apéndice. La llegada del mensaje de iniciación de este protocolo aparece en el *pick* del bloque *exception* en la especificación ACSL correspondiente al rol *participante* del protocolo *IteratedContractNet*. de modo que la iniciación del protocolo de compensación por parte de este agente pueda ser asíncrona respecto de su patrón principal de intercambio.

4.4.5. Asociación de propiedades a los intercambios de mensajes

Los mensajes intercambiados en una conversación llevan información de especial interés en el protocolo que especifica dicha conversación. Así, por ejemplo, los mensajes intercambiados en el contexto de un contract-net pueden llevar un número

de orden de servicio que permita a un agente que participa en múltiples instancias simultáneas de dicho protocolo asociar cada mensaje a la instancia a la que pertenece. En el marco de trabajo de FIPA, la cabecera de un mensaje puede llevar un identificador de conversación (*conversationID*), así como información que ayuda a definir un secuenciamiento relativo de los mensajes en la interacción (*reply-with* e *in-reply-to*). Sin embargo, estos campos de la cabecera o envelope resultan excesivamente generales. Piénsese también que no es un lugar adecuado para llevar información sensible que no se desee permitir ver en el tránsito del mensaje. Si el contenido del mensaje viaja cifrado, por ejemplo, no es conveniente extraer de él el número de orden de servicio. Estas propiedades precisan ser definidas e identificadas en las definiciones de los mensajes. ACSL utiliza para ello expresiones de camino asociadas a elementos *messagePropertyRef*, tal y como se muestra en la figura 4.6. En la versión actual de ACSL, se asume que el mensaje ACL y el contenido deben estar definidos mediante un XML Schema. Esto excluye mensajes que utilicen otro tipo de representación. Para ser claro, el formato verdadero de los mensajes puede ser no-XML, e.g. la representación textual utilizada en la figura 4.4. La semántica canónica de la extracción de una propiedad desde un mensaje implica la traducción del mensaje a un documento XML de acuerdo a los esquemas, y la aplicación a la instancia de una consulta XPath equivalente a la expresión de camino.

Por otra parte, en las figuras 4.7 y 4.8 que aparecen en la sección 4.5, puede observarse que ha sido necesario extender la notación AUML para reflejar la semántica del protocolo. Debido al carácter recursivo del protocolo de Sian, por cada solicitud de modificación, ha sido necesario reflejar una *relación de causalidad* entre la proposición referida en los intercambios que ocurren a lo largo de un hilo de interacción y la proposición a que se refiere el intercambio (*exchange*) que provoca la ejecución de dicho hilo. Tal y como se verá más adelante, los intercambios de mensajes declarados en ese hilo incluirán como parte de su declaración un conjunto de referencias a los parámetros declarados en (o heredados por) el hilo de interacción en que aparecen, significando que dichos parámetros toman valor en función del mensaje recibido o que deben ser utilizados para componer el mensaje enviado, según se trate de una recepción o de un envío, respectivamente. En ambos casos, se utilizan propiedades *messagePropertyRef* para relacionar los parámetros con la información transportada en el mensaje, si bien no es necesario realizar esta asociación para estudiar las propiedades semánticas del protocolos, tal y como se verá en la sección 4.4.5.1.

Las propiedades pueden ser simples o estructuradas. Las propiedades simples se utilizan tanto para correlación como para especificación semántica, mientras que las estructuradas se utilizan para transformar referencias a participantes en la cons-

trucción de topologías dinámicas. Así, como se comentó anteriormente, el elemento *endPointInfPropertyRef* permite obtener de los mensajes entrantes las direcciones a las que deben enviarse los siguientes mensajes. Además, el estado de la interacción en curso está determinado implícitamente/explícitamente a partir de los valores de las propiedades del mensaje entrante. Esto facilita a un agente participar simultáneamente en múltiples instancias de interacciones, puesto que no liga el estado de interacción al estado de ejecución del agente.

4.4.5.1. Especificación en ACSL de la semántica de un protocolo

A la vista de una especificación ACSL de una conversación descrita en AUML, es posible estudiar propiedades sintácticas tales como la alcanzabilidad de un determinado *threadOfInteraction*. Sin embargo, no es posible estudiar propiedades semánticas tales como la terminación del diálogo en tiempo finito. Así, el estudio de las propiedades sintácticas del protocolo 4.2 no determina la terminación del protocolo ya que los intercambios *cfp* y *counter-propose* constituyen un lazo infinito. Tampoco se expresan propiedades semánticas importantes en la determinación de la semántica del protocolo como la relación existente entre las hipótesis que motivan la interacción.

En las figuras 4.2 y 4.3 puede observarse que ha sido necesario extender la notación AUML para reflejar la semántica del protocolo. Debido al carácter recursivo de ambos protocolos, por cada solicitud de propuestas y por cada contrapropuesta, respectivamente, ha sido necesario reflejar una *relación de causalidad* entre la proposición referida en los intercambios que ocurren a lo largo de un hilo de interacción y la proposición a que se refiere el intercambio (*exchange*) que provoca la ejecución de dicho hilo.

ACSL refleja la relación de causalidad entre hilos de interacción mediante *conjuntos de parámetros*, tal y como muestra la figura 4.6. Para ello, la definición de un hilo de interacción incluye, además del propio patrón de intercambio asociado a dicho hilo, la declaración de un conjunto de parámetros encargados de reflejar la parte de semántica del protocolo implicada por dicho hilo. Los hilos de interacción declarados en el cuerpo heredarán todos estos parámetros y podrán además definir otros adicionales. Los intercambios de mensajes declarados en dicho hilo incluirán como parte de su declaración un conjunto de referencias a los parámetros declarados en (o heredados por) el hilo de interacción en que aparecen, significando que dichos parámetros toman valor en función del mensaje recibido o que deben ser utilizados para componer el mensaje enviado, según se trate de una recepción o de un envío, respectivamente. Del mismo modo, estos parámetros pueden aparecer referenciados

en la condición de una construcción *switch* y en la condición *times* de una construcción *pick*, significando que el valor de la condición es función del valor de dichos parámetros.

La cabecera de un protocolo de interacción incluye la declaración de todos los parámetros utilizados en dicho protocolo.

En la especificación ACSL del protocolo de negociación propuesto recogida en el apéndice C puede apreciarse la declaración de los parámetros (construcción *paramSetDelc*, la ulterior referencia a éstos parámetros (construcción *paramSet Ref*) y la instanciación de los parámetros de otros hilos de interacción referenciados con los valores de éstos (construcción *paramSetInst*).

La especificación de ACSL recogida en el apéndice B admite los siguientes tipos de parámetros:

Listas Una lista es un conjunto ordenado de parámetros sobre el que se define un conjunto de operaciones. [] denota la lista vacía, p::l denota una lista no vacía cuyo primer elemento es p y cuyos restantes elementos constituyen a su vez una lista l. Se admiten los siguientes predicados sobre listas:

- $contains(l, e)$: Pertenencia de un elemento e a una lista l.
- $equals(l_1, l_2)$: Igualdad entre listas [ordenadas].
- $equalsSets(l_1, l_2)$: Igualdad entre conjuntos.
- $sublist(l_1, l_2)$: Inclusión de listas.
- $subset(l_1, l_2)$: Inclusión de conjuntos.

y las siguientes funciones:

- $union(l_1, l_2)$: Concatenación de listas.
- $unionSets(l_1, l_2)$: Unión de conjuntos.
- $intersec(l_1, l_2)$: Intersección de listas.
- $intersecSets(l_1, l_2)$: Intersección de conjuntos.
- $difference(l_1, l_2)$: Diferencia de listas.
- $differenceSets(l_1, l_2)$: Diferencia de conjuntos.
- $disjointUnion(l_1, l_2)$: Unión disjunta de listas.
- $disjointUnionSets(l_1, l_2)$: Unión disjunta de conjuntos.

Estos predicados y funciones se utilizan para construir las expresiones que aparecen como condición (*times* o *condition*) de determinadas construcciones

del lenguaje y para construir las expresiones que aparecen en la instanciación de parámetros (*paramSetInst*) asociada a un *threadOfInteractionRef* o a un *protocolRef*.

Naturales Junto con las listas, la utilización de naturales como parámetros facilita la expresión de la semántica asociada al concepto de iteración en construcciones como *pick* o *while* y también la expresión de condiciones en construcciones *switch*. Con el fin de poder utilizar los naturales como parámetros en este tipo de construcciones, se define el conjunto de los números naturales N a partir de la constante $Zero \in N$ y las operaciones $suc(n)$ y $pred(n) / n \neq Zero$ sobre naturales $n \in N$, de modo que $n \in N \leftrightarrow n = Zero \mid \exists n' \in N / n = suc(n')$ constituye una definición inductiva de *natural*. Se cumple que $\forall n \in N, suc(n) = n' \leftrightarrow n = pred(n')$. Se contemplan las siguientes funciones sobre naturales $dec(n_1, n_2)$, $inc(n_1, n_2)$. Son entonces parámetros válidos $Zero$, $suc(n)$, $suc(suc(Zero))$ o $pred(n)$, y son válidas expresiones como $dec(n, suc(Zero))$.

Identificadores de objetos Puede tratarse de proposiciones, creencias, deseos, intenciones, acciones, obligaciones, etc. Se contemplan como objetos de aplicación, por lo que en la construcción de expresiones se admiten referencias a campos o métodos declarados en su correspondiente ontología. Así, si el parámetro m representa un mensaje según la ontología FIPA-ACL, son expresiones válidas $m.reply-to$, $m.from$ o $m.ontology$.

Los dos primeros tipos de parámetros (listas y naturales) permiten realizar ajuste de patrones, mientras que los identificadores de objetos son específicos de la aplicación y su igualdad debe ser evaluada según su semántica. Las especificaciones ACSL recogidas en el apéndice C muestran la utilización de estos parámetros y de sus predicados y funciones asociadas.

4.4.5.2. Correlación de mensajes

Un agente puede, en general, participar simultáneamente en más de una interacción, entendida ésta como un conjunto de conversaciones *interrelacionadas*, bien por entrelazado, bien por anidamiento, especificadas mediante los correspondientes protocolos de interacción. Se hace necesario por tanto disponer de un mecanismo de correlación que permita a los agentes relacionar entre sí los mensajes pertenecientes a una misma interacción, de modo que cada agente pueda determinar a qué conversación pertenecen y sepa relacionarlos con el resto de conversaciones que forman parte de dicha interacción, a la vez que diferenciarlos de los mensajes intercambiados en el contexto de otras interacciones. Con este fin, los diseñadores de sistemas

multi-agente pueden optar entre hacer uso de sofisticadas infraestructuras de transporte que mantengan el estado de las comunicaciones, o bien por implementar ellos mismos un mecanismo de correlación independiente de la infraestructura de transporte utilizada. La segunda de las opciones es más general, flexible y escalable. En el paradigma de objetos distribuidos, pueden encontrarse tanto soluciones que siguen la primera opción como CORBA, DCOM, etc., como soluciones que siguen la segunda opción, como los Servicios Web XML.

Entre las hipótesis asumidas en el diseño de ACSL con el propósito de facilitar que en la especificación de un protocolo de interacción se realicen las menos presunciones posibles acerca del diseño interno de los agentes que la utilizan, se incluye la asunción de que los agentes no utilicen necesariamente una infraestructura de transporte con estado. Se hace por tanto necesaria la especificación del *estado conversacional* en el protocolo de interacción. Por [mantenimiento del] *estado conversacional* se entiende la capacidad de un agente para relacionar entre si todos los mensajes pertenecientes a una misma interacción (mantenida con uno o más agentes y que implica uno o más protocolos de interacción). Esta hipótesis facilita el diseño de agentes más flexibles y escalables ya que les permite, como se verá en el capítulo "Modelo de comunicación", utilizar como "protocolo de transporte" un protocolo sin estado como HTTP. Si los agentes utilizan un protocolo de transporte con estado, simplemente se abstraerán de la información de estado conversacional incluida en la especificación de los protocolos de interacción que utilicen para comunicarse.

A modo de ejemplo, suponga una interacción multiagente en la que inicialmente un agente cliente conversa con un agente broker para negociar la provisión de un servicio haciendo uso del protocolo *iterated-contract-net* y éste último conversa simultáneamente con un grupo de agentes servidores haciendo uso del protocolo *contract-net* para decidir quién proporcionará finalmente el servicio y bajo qué condiciones. El agente broker deberá entrelazar ambas conversaciones de modo que la negociación con el cliente considere las condiciones de servicio ofertadas por los agentes servidores y, al mismo tiempo, la contratación del servicio sea en los términos establecidos por el cliente ya que es éste y no el broker quien deberá fijar las condiciones de servicio. La correlación con el cliente podría ser en base a un identificador de servicio y la correlación con los servidores podría utilizar este mismo token u otro, en cuyo caso habría que relacionar ambos. Posteriormente, el agente que ha proporcionado el servicio deberá facturarlo al broker y este, a su vez, deberá facturar dicho servicio al agente cliente. La correlación servicio-broker precisará tanto del identificador de servicio como de un número de factura, mientras que la correlación broker-cliente precisará tanto del identificador de servicio como de otro

número de factura diferente.

ACSL contempla los escenarios de correlación proporcionando un mecanismo general de especificación que permite definir grupos de mensajes correlacionados a dos niveles: a nivel de instancia de un protocolo de interacción y a nivel de interaccón. En el primer caso, el grupo de correlación está formado por todos los mensajes intercambiados en el contexto de la ocurrencia del protocolo y permite asociar todos estos mensajes a dicha ocurrencia. En el segundo caso, el grupo de correlación está formado por mensajes intercambiados en las múltiples instancias entrelazadas de protocolos que configuran una interacción y permite asociar todos estos mensajes a dicha interacción.

Un *conjunto de correlación* se define como un conjunto nombrado de propiedades (*tokens de correlación*) compartidas por un conjunto de mensajes pertenecientes a una misma interacción. El conjunto de mensajes correlacionado mediante un conjunto de correlación se denomina *grupo de correlación*.

Un mismo token de correlación puede pertenecer a más de un conjunto de correlación (y por tanto a más de un grupo de correlación). Del mismo modo, un mensaje (su declaración *exchange*) puede participar en múltiples grupos de correlación portando por tanto los tokens de los correspondientes conjuntos de correlación. En una instancia de un grupo de correlación, los valores de las propiedades en el conjunto de correlación asociado deben coincidir para todos los mensajes intercambiados en el contexto de la instancia. Cuando una conversación que implica múltiples mensajes se asocia con un conjunto de correlación, todos los mensajes implicados en la definición de esa conversación deben poseer todas las propiedades en el conjunto.

Cuando se describe un intercambio (*exchange*) que referencia un mensaje que tiene asociada información de correlación, la descripción del intercambio lo indica mediante una referencia a un conjunto de correlación.

Los conjuntos de correlación se instancian y terminan dentro del alcance de la instancia de la conversación a la que pertenecen. Un conjunto puede pasar por varias instanciaciones durante la vida de una interacción. La instanciación de un conjunto de correlación se produce con el intercambio de un mensaje del correspondiente grupo de correlación marcado a tal efecto. La vida de una instancia viene determinada por el alcance léxico (contexto) en el que se declara.

En el contexto de una interacción multiparte compuesta por varias conversaciones entrelazadas, cada agente implicado actúa bien como *iniciador*, bien como *seguidor* dentro de una instancia de conversación dada. El iniciador realiza siempre el intercambio del primer mensaje perteneciente al grupo de correlación (que no tiene por qué ser el primer intercambio en la conversación) y por tanto define los valores

reales de las propiedades del conjunto de correlación que etiqueta el protocolo de interacción que especifica la conversación. Los restantes participantes son seguidores que instancian sus conjuntos de correlación en la conversación a partir de los valores de las propiedades presentes en los mensajes entrantes pertenecientes al grupo de correlación. Tanto el iniciador como los seguidores deben marcar el primer mensaje en sus respectivos grupos de correlación como aquel que comienza una instancia del grupo y que por tanto define los valores del conjunto de correlación asociado a dicho grupo.

4.5. Verificación y evaluación semántica de especificaciones ACSL

La especificación de la sintaxis del lenguaje ACSL mediante XML Schema permite validar sintácticamente las especificaciones de conversaciones realizadas mediante dicho lenguaje. Sin embargo, la validación y la evaluación de protocolos de interacción requieren de la consideración tanto de propiedades sintácticas como de propiedades semánticas. Estas últimas implican un subconjunto del entorno en el que actúan los agentes, así como algunas de las acciones realizadas en dicho entorno como consecuencia del transcurso de la conversación. Entre las acciones consideradas se incluyen los propios mensajes intercambiados, en la línea propuesta por la teoría de actos hablados que contempla éstos como acciones. Se requiere por tanto de un formalismo adicional al XML Schema que permita validar y evaluar semánticamente las especificaciones ACSL. Las características del lenguaje ACSL han conducido a la utilización de la Semántica Operacional como aproximación para la especificación del significado dinámico de los protocolos especificados mediante el mismo. La semántica dinámica de un protocolo contempla la ejecución de su especificación, incluida la evaluación de expresiones, el envío y recepción de mensajes y la ejecución de otras acciones no comunicativas.

4.5.1. Desarrollo de una semántica operacional para ACSL

El concepto Semántica Operacional Estructural (SOE) [Plo81, Hen90] denota un formalismo que permite especificar el significado de un lenguaje mediante transformaciones sintácticas de los programas escritos en dicho lenguaje. En esta sección se detalla el proceso de obtención de una semántica operacional adecuada para el lenguaje ACSL basada en la semántica operacional descrita en [EBHM00] y consistente en un sistema transicional de reescritura de términos.

La semántica operacional desarrollada permite la descripción sin ambigüedades

del significado dinámico de las diferentes construcciones de la especificación ACSL de una conversación, facilitando con ello las tareas de análisis, simulación y verificación de dicha especificación. Se ha cuidado especialmente la notación empleada, con el objetivo de hacer más intuitiva la especificación y facilitar una descripción detallada del significado dinámico de características del lenguaje tales como la sincronización de hilos de interacción, la espera por n mensajes, etc. Se presenta también el proceso de obtención de la semántica operacional de una especificación ACSL. Tanto la semántica operacional como el proceso de obtención de la misma se especifican para la sintaxis abstracta de ACSL descrita en el apéndice C.

La aplicación del formalismo de semántica operacional estructurada, creado para lenguajes de programación, al lenguaje de especificación ACSL requiere de una serie de consideraciones. En las siguientes secciones se define la semántica operacional de una especificación ACSL haciendo especial hincapié en estas consideraciones. El proceso de definición se ha dividido en dos partes: definición del sistema transicional y definición del intérprete para dicho sistema, tal y como se propone en [Plo81].

4.5.2. Definición del sistema transicional

Con el fin de obtener un *sistema transicional terminal etiquetado* $\langle \Gamma, \Lambda, \rightarrow, \Upsilon \rangle$ adecuado para ACSL se establece:

$$\Gamma \subseteq \Sigma \times \Theta$$

De modo que una configuración $\gamma = \langle \sigma, \theta \rangle \in \Gamma$ está formada por un identificador de hilo de interacción $\sigma \in \Sigma$ y el conjunto de parámetros $\theta \subseteq \Theta$ que describen el contexto de ejecución para dicho hilo, siendo Σ el conjunto de hilos de interacción declarados en la especificación ACSL de una conversación y Θ el conjunto de parámetros declarados en dichos hilos. En otras palabras, una configuración γ describe un estado conversacional del protocolo especificado, Σ constituye el alfabeto de estados conversacionales y Θ un conjunto de parámetros [de ajuste] para dichos estados.

Del mismo modo, el conjunto de etiquetas Λ se establece como el conjunto de pares $\langle \phi, m \rangle$ formados por el producto cartesiano del conjunto finito de intercambios de mensajes $m \in M$ que ocurren en la especificación ACSL con el conjunto de predicados acerca del entorno $\phi \in \Phi(\omega)$, a partir de los cuales se expresan las condiciones de construcciones del lenguaje como *pick*, *while* o *switch* (y por tanto los parámetros declarados mediante la construcción *paramSetDecl*). Algunos de estos predicados se refieren a los mensajes intercambiados (aparecen en elementos *paramSetRef* asociados a elementos *exchange*), por lo que se denotan como $\phi(m)$ y se asocian a los mensajes intercambiados con el fin de explicitar esta relación.

La relación de transición

$$\rightarrow \subseteq \Gamma \times \Lambda \times \Gamma$$

queda entonces como

$$\rightarrow \subseteq (\Sigma \times \Theta) \times (\Phi \times M) \times (\Sigma \times \Theta)$$

La relación $\rightarrow$ representa por tanto una relación de transición entre estados conversacionales etiquetada mediante una acción. La idea, común a otros sistemas transicionales etiquetados, es que la acción asociada a una transición proporciona información acerca de qué sucede en la configuración durante la transición (acciones internas) y/o acerca de la interacción entre los agentes y su entorno (acciones externas). En este caso, las acciones se refieren a la comunicación entre los agentes, por lo que la información suministrada son los propios mensajes intercambiados y la información de configuración (parámetros de los estados conversacionales) implicada por éstos.

Así, el alfabeto del lenguaje está constituido por el conjunto de posibles mensajes intercambiados en el transcurso de una conversación $M = \{m_i, i = 1..n\}$, de modo que dicho lenguaje se puede definir como el conjunto de posibles secuencias de mensajes intercambiados, cada una de las cuales constituye una palabra del lenguaje:

$$L \equiv \varepsilon \in L \mid m \in M \rightarrow m \in L \mid s_1, s_2 \in L' \rightarrow s_1 \odot s_2 \in L'$$

Por último, el conjunto de configuraciones terminales $\Upsilon \subseteq \Gamma$ viene determinado por aquellas configuraciones con hilos de interacción en los que se intercambia un mensaje etiquetado como *terminal*, ya que éstas son las únicas para las que se cumple

$$\forall \gamma \in \Upsilon, \forall \gamma' \in \Gamma \cdot \gamma \nrightarrow \gamma'$$

Este sistema transicional presentado hasta el momento está basado en la definición aportada por Plotkin. Sin embargo, esta definición puede extenderse con el conjunto de acciones $\alpha \subseteq \Lambda(\omega)$ a ejecutar en cada transición. La relación de transición quedaría entonces:

$$\rightarrow \subseteq (\Sigma \times \Theta) \times (\Phi \times M) \times (\Sigma \times \Theta) \times \alpha \subseteq \Lambda(\omega)$$

o bien

$$\rightarrow \subseteq \Gamma \times \Lambda \times \Gamma \times \Lambda$$
$$\rightarrow \subseteq (\Sigma \times \Theta) \times (\Phi \times M) \times (\Sigma \times \Theta) \times (\Lambda)$$

4.5.3. Definición del intérprete

Siguiendo a [Hen90], el formalismo SOE permite construir la semántica operacional de una especificación ACSL mediante la descripción formal de un intérprete I de dicho lenguaje cuyo comportamiento viene especificado por un conjunto de reglas de producción.

I se modela como una función que toma como argumento un protocolo P especificado en el lenguaje ACSL, un entorno ω, y describe el comportamiento de $\langle P, \omega \rangle$ como una serie [in]finita de producciones del estilo $\langle P, \omega \rangle \rightarrow \langle P_1, \omega_1 \rangle \rightarrow \langle P_2, \omega_2 \rangle \rightarrow \ldots$ Si P termina, entonces el resultado es $\langle END, \omega_n \rangle$.

Así, la especificación del autómata que actúa como intérprete de un protocolo de interacción, y que por tanto determina la semántica operacional de una especificación ACSL, define un conjunto de reglas de producción que constituye la definición de dicho intérprete, mientras que la secuencia de mensajes enviados y recibidos constituye el programa que debe ser interpretado. Suponiendo que los agentes presentan una arquitectura interna BDI, el conjunto de creencias (β), deseos (δ) e intenciones (ι) del agente constituye el entorno ω y predicados acerca del mismo (e.g. WantToPropose(p), IntendToDo(t), etc.), con lo que $\Phi(\omega) \subseteq \beta \cup \delta \cup \iota$ y el conjunto de acciones (incluidos los mensajes intercambiados) constituyen los efectos laterales sobre ω, del mismo modo que una asignación de variable en un lenguaje de programación.

Como se ha dicho, la semántica operacional desarrollada está basada en un sistema de producción que mapea estados conversacionales a nuevos estados conversacionales, para una especificación ACSL dada. Una regla de producción adopta el formato que se muestra a continuación:

Este sistema transicional puede verse como un sistema de producción en el que cada transición viene determinada por el disparo de una regla ante la ocurrencia de una acción y la validez del predicado ϕ. Escribiremos entonces una transición como.

$$\begin{array}{l} e \in \Sigma \\ \theta \in \Theta \\ \phi \in \Phi(\omega) \\ \rightarrow msg/\phi(msg) \\ \hline \\ e' \in \Sigma \\ \theta' \in \Theta \\ \alpha/\alpha \in \Lambda(\omega) \end{array} \text{ regla x}$$

que, de forma más compacta puede expresarse como

$$\langle\langle e \in \Sigma, \theta \in \Theta\rangle, \phi \in \Phi(\omega)\rangle \xrightarrow[msg/\phi(msg)]{} \langle\langle e' \in \Sigma, \theta' \in \Theta\rangle, [\alpha]/\alpha \in \Lambda(\omega)\rangle$$

Esta acción puede representar el envío de un mensaje, su recepción, o una acción interna del agente, por lo que se consideran por tanto tres tipos de reglas de producción:

Tipo 1. Regla de producción disparada por el envío de un mensaje, siendo cierto ϕ:

$$\langle\langle e \in \Sigma, \theta \in \Theta\rangle, \phi \in \Phi(\omega)\rangle \xrightarrow[msg/\phi(msg)]{} \langle\langle e' \in \Sigma, \theta' \in \Theta\rangle, \alpha \in \Lambda(\omega)\rangle$$

Tipo 2. Regla de producción disparada por la recepción de un mensaje, siendo cierto ϕ:

$$\langle\langle e \in \Sigma, \theta \in \Theta\rangle, \phi \in \Phi(\omega)\rangle \xrightarrow{msg/\phi(msg)} \langle\langle e' \in \Sigma, \theta' \in \Theta\rangle, \alpha \in \Lambda(\omega)\rangle$$

Tipo 3. Regla de producción disparada por una acción interna del agente, siendo cierto ϕ:

$$\langle\langle e \in \Sigma, \theta \in \Theta\rangle, \phi \in \Phi(\omega)\rangle \overrightarrow{\tau}_{\varepsilon} \langle\langle e' \in \Sigma, \theta' \in \Theta\rangle, \alpha \in \Lambda(\omega)\rangle$$

En todas ellas se ha calificado la relación de transición con el envío $\xrightarrow[msg/\phi(msg)]{}$ o la recepción $\xrightarrow{msg/\phi(msg)}$ de un patrón de mensaje $msg/\phi(msg)$. El formato de msg es idéntico al utilizado para representar una configuración parametrizada $\langle m \in M, \theta \in \Theta\rangle$, siendo M el alfabeto de performativas y θ una tupla de parámetros de ajuste para el mensaje.

El predicado $\phi(msg)$ acerca de los mensajes recibidos permite, por ejemplo, averiguar si el origen del mensaje ya envió otro con anterioridad (habría que considerar también la pertenencia del origen al grupo de agentes que participan en el protocolo). Este predicado representa la parte del predicado que aparece en la premisa directamente relacionada con el mensaje.

$$\phi(msg) = true \leftrightarrow \neg\exists m \in msgQ / m.from = msg.from$$

La descripción parametrizada de un estado conversacional del protocolo $\langle e \in \Sigma, \theta \in \Theta\rangle$, consiste en un identificador de estado e perteneciente al alfabeto de estados Σ y una tupla θ de parámetros de ajuste para el estado e. La tupla de parámetros de ajuste incluye (1) variables de tipo Int, Char, List, Tuple orientadas al ajuste de patrones o (2) variables orientadas a la representación de creencias, deseos o intenciones.

$\Phi(\omega)$ representa el conjunto de predicados acerca del entorno que pueden ser evaluados por los agentes que participan en la conversación. Estos predicados suelen estar intrínsecamente relacionados con las creencias, deseos e intenciones de los

agentes. Sin embargo, no se asume nada acerca del modo en que los agentes llevan a cabo su evaluación ya que ésta no sólo puede estar relacionada con los estados conversacionales del protocolo, sino que también puede estar relacionada con el estado interno de los agentes.

Durante la interpretación de un programa (constituido por una serie de mensajes) no existe ninguna forma de conocer con antelación el programa final. Además, no puede conocerse la secuencia de mensajes que serán recibidos. Esto muestra la adecuación de la filosofía SOE para la descripción de la semántica de este tipo de programa. De hecho, una de sus características más sobresalientes es que trabaja desde el pasado hacia el futuro (hecho característico de una semántica directa) y que el significado del programa se construye a medida que se interpreta el texto del programa. Al contrario de lo que ocurre con otras filosofías como la semántica por continuación, característica de los compiladores, en la que el significado del programa se construye antes de su ejecución.

4.5.4. Generación del intérprete a partir de una especificación ACSL

Las siguientes subsecciones detallan el proceso de generación del intérprete para cada una de las construcciones del lenguaje ACSL.

4.5.4.1. Simplificaciones

Con el fin de simplificar la generación de reglas de producción en construcciones anidadas, se utiliza la siguiente regla de simplificación:

Si se tienen dos reglas

$$\langle\langle A \in \Sigma, \theta \in \Theta\rangle, \phi \in \Phi(\omega)\rangle \xrightarrow[msg/\phi(msg)]{} \langle\langle B \in \Sigma, \theta' \in \Theta\rangle, \alpha \in \Lambda(\omega)\rangle$$

y

$$\langle\langle B \in \Sigma, \theta' \in \Theta\rangle, \phi' \in \Phi(\omega)\rangle \overrightarrow{\varepsilon} \langle\langle C \in \Sigma, \theta'' \in \Theta\rangle, \alpha' \in \Lambda(\omega)\rangle$$

pueden simplificarse por

$$\langle\langle A \in \Sigma, \theta \in \Theta\rangle, \phi \in \Phi(\omega)\rangle \xrightarrow[msg/\phi(msg)]{} \langle\langle C \in \Sigma, \theta'' \in \Theta\rangle, \alpha' \in \Lambda(\omega)\rangle$$

Siempre que $\phi \rightarrow \phi'$.

Idem para las reglas de envío de mensajes.

4.5.4.2. Reglas de producción para opcionalidad

ACSL permite expresar opcionalidad en el curso de una conversación mediante la construcción *switch*. Se muestra a continuación la estructura general de esta construcción, tal y como se recoge en la sintaxis abstracta del lenguaje detallada en el apéndice C:

```
Switch := switch Multichoice {Branch} [Default]
Multichoice := multichoice boolean
Branch := branch Case ActionBlock
Default := default (ThreadOfInteraction | ProcolControlGroup | Action)
Case := case Condition [ParamSetRef]
Condition := condition [ParamSetRef] Expression
ActionBlock := action (ThreadOfInteraction | ProtocolControlGroup | Action)
ParamSetRef := paramSetRef {ParamRef}
ParamRef := paramRef Mode string
Mode := match | adjust
```

Para cada una de las ramas *branch* se obtiene el siguiente patrón de regla:

$$\langle\langle A \in \Sigma, \{\theta_i \in \Theta\}\rangle, \phi \in \Phi(\omega)\rangle \overrightarrow{\varepsilon} \langle\langle B_j \in \Sigma, \{\theta'_k \in \Theta\}\rangle, [\alpha \in \Lambda(\omega)]\rangle$$

En la que $A \in \Sigma$ denota el estado conversacional generado para el switch, $\{\theta_i \in \Theta\}$ es la lista de parámetros referenciados (*paramSetRef*) en la condición de la rama considerada, $\phi \in \Phi(\omega)$ es la propia condición, $B_j \in \Sigma$ denota un nuevo estado conversacional que será utilizado en el antecedente de las reglas generadas para el hilo de interacción que define la acción de dicha rama. Si se trata de una referencia a un hilo de interacción, $\{\theta'_k \in \Theta\}$ será el conjunto de valores con que se instancian los parámetros de dicho hilo (*paramSetInst*).

Para la rama *default* se obtiene el mismo patrón de regla considerando:

$$\phi = \bigcup_{i=1}^{n} \phi_i \qquad \text{y} \qquad \theta = \neg \bigvee_{i=1}^{n} \theta_i$$

4.5.4.3. Reglas de producción para la recepción de n mensajes

La espera por la recepción de n mensajes provenientes de diferentes agentes se expresa mediante la instrucción pick:

```
Pick := pick [Times] [ParamSetRef] {EventHandler} [OnTimes]
EvenHandler := eventHandler Event ActionBlock
OnTimes := onTimes (ThreadOfInteraction | ProtocolControlGroup | Action)
EventHandler := eventHandler Event ActionBlock
Event := event Id (DelayFor | DelayUntil | Exchange | Catch)
ActionBlock := action (ThreadOfInteraction | ProtocolControlGroup | Action)
Times := Expression
Id:= id string
```

para la cual se obtiene el siguiente conjunto patrón de reglas de producción por cada uno de los manejadores *EventHandler*:

$$\langle\langle A \in \Sigma, (\dots Succ(t) \dots)\rangle, true\rangle \xrightarrow[msg_1/\phi(msg_1)]{} \langle\langle A, (\dots (t) \dots)\rangle, [\alpha \in \Lambda(\omega)]\rangle\rangle$$
$$\langle\langle A \in \Sigma, (\dots Succ(t) \dots)\rangle, true\rangle \xrightarrow[msg_2/\phi(msg_2)]{} \langle\langle A, (\dots (t) \dots)\rangle, [\alpha \in \Lambda(\omega)]\rangle\rangle$$
$$\dots$$
$$\langle\langle A \in \Sigma, (\dots Zero \dots)\rangle, true\rangle \xrightarrow[\varepsilon]{} \langle\langle B \in \Sigma, \{\theta \in \Theta\}\rangle, [\alpha \in \Lambda(\omega)]\rangle\rangle$$

en el que las expresiones $(\dots Succ(t) \dots)$ incluyen todos los parámetros referenciados en el cuerpo del pick (incluidos sus sucesos) y la expresión $(\theta \in \Theta)$ estará compuesta por el conjunto de valores con que se instancian los parámetros del hilo referenciado en la expresión <onTimes>.

Se ha supuesto que todos los manejadores del pick son por la espera de un mensaje. En otro caso, la condición de espera del manejador se reflejaría en $\phi \in \Phi(\omega)$, que dejaría de ser *true* y la regla de transición pasaría a ser $\xrightarrow[\varepsilon]{}$.

Así, para el siguiente pick:

```
<pick times="length(apl)">
  <paramSetRef>
     <paramRef mode="match">apl</paramRef>
  </paramSetRef>
  <eventHandler>
     <event>
        <exchange message="Inform" direction="in" mode="middle">
           <paramSetRef>
              <paramRef mode="adjust">p(t1)-->ipl</paramRef>
              <paramRef mode="match">t1</paramRef>
           </paramSetRef>
        </exchange>
     </event>
     <action>
        <empty/>
     </action>
  </eventHandler>
  <eventHandler>
     <event>
        <exchange message="Failure" direction="in" mode="middle">
           <paramSetRef>
              <paramRef mode="adjust">p(t1)-->fpl</paramRef>
              <paramRef mode="match">t1</paramRef>
           </paramSetRef>
        </exchange>
     </event>
     <action>
       <empty/>
     </action>
  </eventHandler>
  <onTimes>
     <threadOfInteractionRef threadRef="End">
        <paramSetInst>
```

```
            <paramInst>
               <ref>t1</ref>
               <value>t1</value>
            </paramInst>
            <paramInst>
               <ref>pl</ref>
               <value>fpl</value>
            </paramInst>
         </paramSetInst>
      </threadOfInteractionRef>
   </onTimes>
</pick>
```

se obtendrían las siguientes reglas:

$$\langle\langle A \in \Sigma, (t_1, Succ(n), fpl, ipl)\rangle, true\rangle \xrightarrow[Fail(p(t_1))]{} \langle\langle A, (t_1, n, p :: fpl, ipl)\rangle, NOOP)\rangle$$

$$\langle\langle A \in \Sigma, (t_1, Succ(n), fpl, ipl)\rangle, true\rangle \xrightarrow[Inform(p(t_1))]{} \langle\langle A, (t_1, n, fpl, p :: ipl)\rangle, NOOP)\rangle$$

$$\langle\langle A \in \Sigma, (T_1, Zero, fpl, ipl)\rangle, true\rangle \overrightarrow{}_{\varepsilon} \langle\langle B \in \Sigma, t_1, fpl\rangle, NOOP)\rangle$$

4.5.4.4. Reglas de producción para iteraciones

Las iteraciones se expresan en ACSL mediante la instrucción while:

```
While := while Condition ActionBlock\\
Condition := condition [ParamSetRef] Expression\\
ActionBlock := action (ThreadOfInteraction | ProtocolControlGroup
| Action)\\
```

para la cual se obtiene el siguiente conjunto patrón de reglas de producción:

$$\langle\langle A \in \Sigma, \theta \in \Theta\rangle, \phi \in \Phi(\omega)\rangle \overrightarrow{}_{\varepsilon} \langle\langle A \in \Sigma, \theta_1 \in \Theta\rangle, \alpha \in \Lambda(\omega)\rangle$$

$$\langle\langle A \in \Sigma, \theta_2 \in \Theta\rangle, \phi \in \Phi(\omega)\rangle \overrightarrow{}_{\varepsilon} \langle\langle B \in \Sigma, \theta_3 \in \Theta\rangle, \alpha \in \Lambda(\omega)\rangle$$

La finalización del while supone la convergencia de Θ a un estado en que θ_2 se haga cierta.

El siguiente ejemplo supone que se desea enviar un mensaje a todos los agentes referenciados en una lista:

$$\langle\langle A \in \Sigma, \ldots p :: l \ldots\rangle, \phi \in \Phi(\omega)\rangle \xrightarrow[msg(p.from)]{} \langle\langle A \in \Sigma, \ldots l \ldots\rangle, \alpha \in \Lambda(\omega)\rangle$$

$$\langle\langle A \in \Sigma, \ldots [] \ldots\rangle, \phi \in \Phi(\omega)\rangle \overrightarrow{}_{\varepsilon} \langle\langle B \in \Sigma, \theta \in \Theta\rangle, \alpha \in \Lambda(\omega)\rangle$$

4.5.4.5. Reglas para composición de estructuras

Se muestran a continuación las reglas resultantes de aplicar los patrones vistos en los apartados anteriores (incluido el patrón de simplificación) a la siguiente especificación ACSL que compone una estructura *while* con una estructura *switch*:

```
<threadOfInteraction>
  <while>
    <condition condition="existProposalInProposals">
      <paramSetRef>
        <paramRef mode="adjust">p(t1)::pl</paramRef>
        <paramRef mode="match">t1</paramRef>
      </paramSetRef>
    </condition>
    <action>
      <switch multiChoice="false">
         <branch>
            <case condition="WantToAcceptProposal">
               <paramSetRef>
                 <paramRef mode="match">p</paramRef>
                 <paramRef mode="adjust">p-->apl</paramRef>
               </paramSetRef>
            </case>
            <action>
               <exchange message="Accept" direction="out" delivery="unreliable"
                         mode="middle" type="asynchronous">
                 <paramSetRef>
                   <paramRef mode="match">p.id</paramRef>
                   <paramRef mode="match">p</paramRef>
                 </paramSetRef>
               </exchange>
            </action>
         </branch>
         <branch>
            <case condition="WantToRejectProposal">
               <paramSetRef>
                 <paramRef mode="match">p</paramRef>
               </paramSetRef>
            </case>
            <action>
               <exchange message="Reject" direction="out" delivery="unreliable"
                         mode="middle" type="asynchronous">
                  <paramSetRef>
                    <paramRef mode="match">p.id</paramRef>
                    <paramRef mode="match">p</paramRef>
                  </paramSetRef>
               </exchange>
            </action>
         </branch>
      </switch>
    </action>
  </while>
```

En un primer paso previo a una posible simplificación se obtiene:

$$\langle\langle \mathrm{A}, t_1, p(t_1) :: pl, apl\rangle, true\rangle \overrightarrow{\varepsilon} \langle\langle \mathrm{B}, t_1, pl, p, apl\rangle, \mathrm{NOOP}\rangle$$

$$\langle\langle \mathrm{B}, t_1, pl, p, apl\rangle, WantToAcceptProposal(p)\rangle \xrightarrow{Accept(p)} \langle\langle \mathrm{A}, t_1, pl, p :: apl\rangle, NOOP\rangle$$

$$\langle\langle \mathrm{B}, t_1, pl, p, apl\rangle, WantToRejectProposal(p)\rangle \xrightarrow{Reject(p)} \langle\langle \mathrm{A}, t_1, pl, apl\rangle, NOOP\rangle$$

$$\langle\langle \mathrm{A}, t_1, [], apl\rangle, true\rangle \overrightarrow{\varepsilon} \langle\langle \mathrm{C}, l\rangle, \mathrm{NOOP}\rangle$$

que simplificando queda como:

$$\langle\langle \mathrm{A}, t_1, p(t_1) :: pl, apl\rangle, WantToAcceptProposal(p)\rangle \xrightarrow{Accept(p)} \langle\langle \mathrm{A}, t_1, pl, p :: apl\rangle, NOOP\rangle$$

$$\langle\langle \mathrm{A}, t_1, p(t_1) :: pl, apl\rangle, WantToRejectProposal(p\rangle \xrightarrow{Reject(p)} \langle\langle \mathrm{A}, t_1, pl, apl\rangle, NOOP\rangle$$

$$\langle\langle \mathrm{A}, t_1, [], apl\rangle, true\rangle \overrightarrow{\varepsilon} \langle\langle \mathrm{C}, l\rangle, \mathrm{NOOP}\rangle$$

Reglas generales:

$$\langle\langle \mathrm{A}, p :: l\rangle, true\rangle \overrightarrow{\varepsilon} \langle\langle \mathrm{B}, l\rangle, \mathrm{NOOP}\rangle$$

$$\langle\langle \mathrm{B}, l\rangle, c_1\rangle \xrightarrow{msg_1} \langle\langle \mathrm{A}, l\rangle, NOOP\rangle$$

$$\langle\langle \mathrm{B}, l\rangle, c_2\rangle \xrightarrow{msg_2} \langle\langle \mathrm{A}, l\rangle, NOOP\rangle$$

$$\langle\langle \mathrm{A}, []\rangle, true\rangle \overrightarrow{\varepsilon} \langle\langle \mathrm{C}, \rangle, \mathrm{NOOP}\rangle$$

simplificando:

$$\langle\langle \mathrm{A}, p :: l\rangle, c_1\rangle \xrightarrow{msg_1} \langle\langle \mathrm{A}, l\rangle, NOOP\rangle$$

$$\langle\langle \mathrm{A}, p :: l\rangle, c_2\rangle \xrightarrow{msg_2} \langle\langle \mathrm{A}, l\rangle, NOOP\rangle$$

$$\langle\langle \mathrm{A}, []\rangle, true\rangle \overrightarrow{\varepsilon} \langle\langle \mathrm{C}, \rangle, \mathrm{NOOP}\rangle$$

4.5.4.6. Reglas para la composición secuencial de sentencias

La transición para la composición secuencial de sentencias $s_1; S$ puede derivarse de la transición para la sentencia s_1.

$$\langle\langle A \in \Sigma, s_1, \theta_1 \in \Theta\rangle, \phi_1 \in \Phi(\omega)\rangle \overrightarrow{\varepsilon} \langle\langle A_1 \in \Sigma, \phi, \theta'_1 \in \Theta\rangle, \alpha_1 \in \Lambda(\omega)\rangle$$
$$\ldots$$
$$\langle\langle A \in \Sigma, s_1, \theta_2 \in \Theta\rangle, \phi_2 \in \Phi(\omega)\rangle \overrightarrow{\varepsilon} \langle\langle A_2 \in \Sigma, \phi, \theta'_2 \in \Theta\rangle, \alpha_2 \in \Lambda(\omega)\rangle$$

$$\langle\langle A \in \Sigma, s_1; S, \theta_1 \in \Theta\rangle, \phi_1 \in \Phi(\omega)\rangle \overrightarrow{\varepsilon} \langle\langle A_1 \in \Sigma, S, \theta'_1 \in \Theta\rangle, \alpha_1 \in \Lambda(\omega)\rangle$$
$$\ldots$$
$$\langle\langle A \in \Sigma, s_1; S, \theta_2 \in \Theta\rangle, \phi_2 \in \Phi(\omega)\rangle \overrightarrow{\varepsilon} \langle\langle A_2 \in \Sigma, S, \theta'_2 \in \Theta\rangle, \alpha_2 \in \Lambda(\omega)\rangle$$

La finalización del while supone la convergencia de Θ a un estado en que θ_2 se haga cierta.

4.5.4.7. Semántica Operacional para la especificación ACSL del protocolo IteratedContractNet

A continuación se muestran las reglas que definen el intérprete que se obtiene siguiendo las reglas anteriores a partir de la especificación ACSL del protocolo IteratedContractNet según el estándar [FIP02].

$$\langle\langle \text{Init}, t_1 \in Tasks, N\rangle, WantProposals(t_1)\rangle \xrightarrow{cfp(t_1)} \langle\langle \text{WaitOpinion}, t_1, N, []\rangle, NOOP\rangle$$

$$\langle\langle \text{WaitOpinion}, t_1, succ(N), l\rangle, true\rangle \xrightarrow[Refuse(t_1)]{} \langle\langle \text{WaitOpinion}, t_1, N, l\rangle, NOOP\rangle$$

$$\langle\langle \text{WaitOpinion}, t_1, succ(N), l\rangle, true\rangle \xrightarrow[Propose(p(t_1))]{} \langle\langle \text{WaitOpinion}, t_1, N, p(t_1) :: l\rangle, NOOP\rangle$$

$$\langle\langle \text{WaitOpinion}, t_1, zero, l\rangle, true\rangle \xrightarrow[Refuse(t_1)]{} \langle\langle \text{OpinionProposals}, t_1, l\rangle, NOOP\rangle$$

$$\langle\langle \text{WaitOpinion}, t_1, zero, l\rangle, true\rangle \xrightarrow[Propose(p(t_1))]{} \langle\langle \text{OpinionProposals}, t_1, p(t_1) :: l\rangle, NOOP\rangle$$

$$\langle\langle \text{WaitOpinion}, t_1, zero, []\rangle, true\rangle \xrightarrow[Refuse(t_1)]{} \langle\langle \text{End}, \rangle, NOOP\rangle$$

$$\langle\langle \text{WaitOpinion}, t_1, zero, []\rangle, true\rangle \xrightarrow[Propose(p(t_1))]{} \langle\langle \text{OpinionProposals}, t_1, p(t_l) :: []\rangle, NOOP\rangle$$

$$\langle\langle \text{OpinionProposals}, t_1, l\rangle, WantToIterate(l, t_2)\rangle \overrightarrow{}_{\varepsilon} \langle\langle \text{IterateProposals}, t_2, N, []\rangle, \text{NOOP}\rangle$$

$$\langle\langle \text{OpinionProposals}, t_1, l\rangle, \sim WantToIterate(l, t_2)\rangle \overrightarrow{}_{\varepsilon} \langle\langle \text{AcceptProposals}, t_1, l, 0\rangle, \text{NOOP}\rangle$$

$$\langle\langle \text{AcceptProposals}, t_1, p(t_1) :: l, N\rangle, WantToAcceptProposal(p(t_1))\rangle \xrightarrow{Accept(p(t_1))} \langle\langle \text{AcceptProposals}, t_1, l, Succ(N)\rangle, NOOP\rangle$$

$$\langle\langle \text{AcceptProposals}, t_1, p(t_1) :: l, N\rangle, \sim WantToRejectProposal(p(t_1))\rangle \xrightarrow{Reject(p(t_1))} \langle\langle \text{AcceptProposals}, t_1, l, Succ(N)\rangle, NOOP\rangle$$

$$\langle\langle \text{AcceptProposals}, t_1, [], N\rangle, true\rangle \overrightarrow{}_{\varepsilon} \langle\langle \text{WaitCompletion}, N\rangle, \text{NOOP}\rangle$$

$$\langle\langle \text{IterateProposals}, t_1, p(t_1) :: l, M\rangle, WantToIterateProposal(p(t_1))\rangle \xrightarrow{cfp(t_1)} \langle\langle \text{IterateProposals}, t_1, l, succ(M)\rangle, NOOP\rangle$$

$$\langle\langle \text{IterateProposals}, t_1, p(t_1) :: l, M\rangle, \sim WantToIterateProposal(p(t_1))\rangle \xrightarrow{Reject(p(t_1))} \langle\langle \text{IterateProposals}, t_1, l, M\rangle, NOOP\rangle$$

$$\langle\langle \text{IterateProposals}, t_1, [], M\rangle, true\rangle \overrightarrow{}_{\varepsilon} \langle\langle \text{WaitOpinion}, t_1, M, []\rangle, \text{NOOP}\rangle$$

$$\langle\langle \text{WaitCompletion}, t_1, succ(N), apl, fpl\rangle, true\rangle \xrightarrow[Inform(p(t_1))]{} \langle\langle \text{WaitCompletion}, t_1, N, p(t_1) :: apl, fpl\rangle, NOOP\rangle$$

$$\langle\langle \text{WaitCompletion}, t_1, succ(N), apl, fpl\rangle, true\rangle \xrightarrow[Failure(p(t_1))]{} \langle\langle \text{WaitCompletion}, t_1, N, apl, p(t_1) :: fpl\rangle, NOOP\rangle$$

$$\langle\langle \text{WaitCompletion}, t_1, Zero, apl, p :: fpl\rangle, true\rangle \overrightarrow{}_{\varepsilon} \langle\langle \text{Compensation}, t_1, fpl\rangle, \text{NOOP}\rangle$$

$$\langle\langle \text{WaitCompletion}, t_1, Zero, apl, []\rangle, true\rangle \overrightarrow{}_{\varepsilon} \langle\langle \text{End}, \rangle, \text{NOOP}\rangle$$

4.5.5. Ejemplo recopilatorio

Los protocolos de interacción estandarizados por FIPA se han mostrado adecuados como prueba de concepto para ilustrar el poder expresivo de la notación gráfica AUML. Sin embargo, adolecen en general del grado de complejidad necesario para mostrar el poder expresivo del lenguaje ACSL. A lo largo de los años se han desarrollado numerosos protocolos de cooperación que, pese a presentar un fuerte componente de interacción y haber sido comúnmente aceptados y utilizados por la comunidad internacional de DAI, no han sido incluidos hasta la fecha en la biblioteca de protocolos de interacción de FIPA. El protocolo de aprendizaje cooperativo desarrollado por Sati Singh Sian [Sia91] constituye un ejemplo paradigmático, por lo que se ha escogido para ilustrar la utilización de ACSL como lenguaje de especificación. El apéndice C recoge la especificación completa ACSL, así como los intérpretes SOA, para las diferentes versiones del protocolo.

Se trata de un protocolo de aprendizaje en el que dos o más agentes dialogan con el fin de alcanzar acuerdos acerca de las diferentes hipótesis elaboradas por cada uno de ellos. A la hora de describir este protocolo conviene diferenciar dos posibles escenarios de aplicación que derivarán en dos especificaciones diferentes: el protocolo seguido por dos agentes y el protocolo seguido por n agentes (para $n > 2$).

4.5.5.1. Descripción de las acciones comunicativas utilizadas

En la descripción del protocolo de coordinación de Singh Sian, se asume la existencia de un conjunto de actos hablados, de mayor nivel que los disponibles en KQML o FIPA-ACL, que definen [un subconjunto de] las acciones comunicativas disponibles en cada uno de los agentes que ofrece este protocolo como interfaz de comunicación. A continuación se describe de manera informal la semántica de cada uno de estos actos hablados (representados como performativas del ACL correspondiente):

4.5.5.2. Descripción del protocolo para dos agentes

Cada uno de los agentes dispone de una base de experiencia B_e y de una base de conocimiento B_c. En el momento en que uno de los dos agentes desee proponer una hipótesis elaborada a partir del conocimiento acumulado en su B_e, iniciará el protocolo enviando al otro agente un mensaje con su propuesta p. El otro agente debe entonces contestar confirmando dicha hipótesis si está de acuerdo con su verdad en función del conocimiento acumulado en su B_e o en su B_c privadas, mostrando su desacuerdo si dispone de una hipótesis opuesta en su B_c, proponiendo una modificación si es capaz de elaborar una propuesta más general o más específica a partir de

su B_c, o indicando su incapacidad para opinar acerca de dicha hipótesis en cualquier otro caso. A partir de la opinión vertida por este último, el agente que formuló la hipótesis inicial evaluará su verdad en función de un determinado criterio preestablecido, pudiendo abandonarla (eliminándola de su B_e) o confirmarla (pasándola a su B_c). La decisión tomada debe ser comunicada al otro agente para que éste, que en caso de confirmación deberá comunicar su aceptación. En caso de que la opinión vertida por el segundo agente sea una propuesta de modificación de la hipótesis inicial por otra nueva hipótesis, se iniciará un proceso isomórfico al presentado en el que el agente que realizó la propuesta inicial deberá opinar acerca de la nueva hipótesis y será el otro agente quien deba evaluar su verdad a partir de la opinión vertida por el primero. Este proceso se repetirá hasta que alguno de los dos agentes cese en sus propuestas de modificación y decida confirmar, oponerse o abstenerse con respecto a la hipótesis en curso.

Al margen de este proceso de aprendizaje cooperativo, cualquiera de los dos agentes podrá aseverar una proposición p a partir del conocimiento acumulado en su B_c, en cuyo caso el otro agente simplemente comunicará su aceptación y la incluirá en su B_c (eliminándola de su B_e si fuese necesario).

La figura 4.7 muestra la representación AUML del protocolo descrito.

4.5.5.3. Descripción del protocolo para n agentes

El protocolo se inicia cuando uno de los agentes A_i participantes desea recabar la opinión del resto acerca de una hipótesis propia p elaborada a partir del conocimiento recopilado en su base de experiencia B_e. El agente iniciador difunde entonces una proposición al resto de agentes que, una vez evaluada, contestarán confirmado dicha hipótesis, mostrando su desacuerdo, señalando su falta de opinión o proponiendo su modificación por otra proposición q más general o más específica, siempre en función del contenido de sus bases de conocimiento. A_i esperará entonces por todas estas respuestas durante un tiempo limitado. En caso de no recibir ninguna petición de modificación, A_i deberá tomar una decisión acerca de la verdad de p y notificársela al resto de agentes confirmándo o retirando la proposición inicial. En caso de recibir alguna petición de modificación, el agente deberá escoger una en base a un criterio preestablecido (que en todo caso no forma parte del protocolo de coordinación) hacerla pública (identificando tanto la propuesta de modificación como el agente que la elaboró) y opinar acerca suyo confirmándola, mostrando su desacuerdo, su falta de opinión o proponiendo una nueva modificación.

El protocolo se complica ante la necesidad de contemplar la situación en que el agente recibe una indicación de una modificación, ya que esta indicación puede

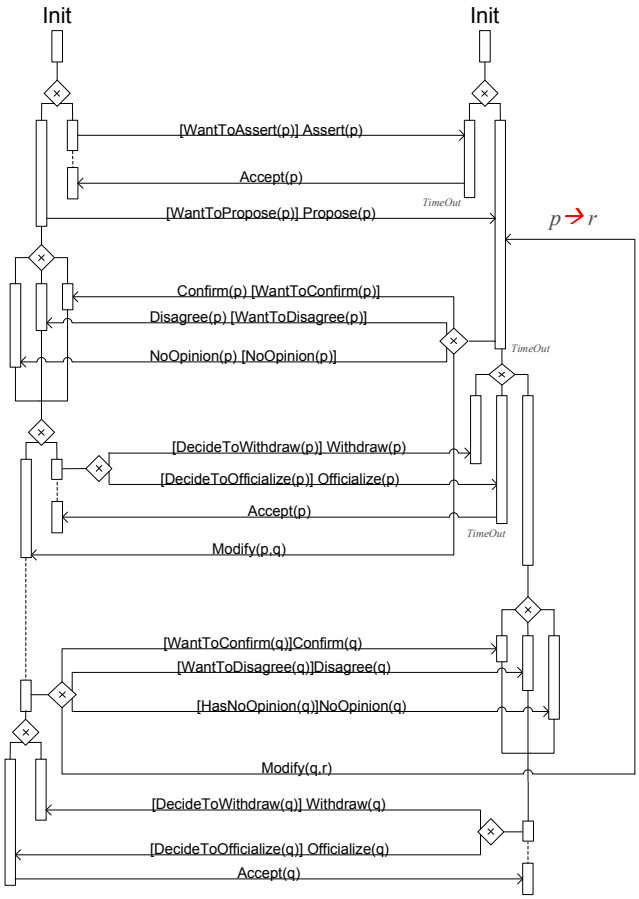

Figura 4.7: Especificación AUML del protocolo de coordinación de Sian para dos agentes

corresponder tanto a una solicitud de modificación elaborada por él mismo; en cuyo caso recabará la opinión del resto de agentes, como a una solicitud de modificación elaborada por otro agente; en cuyo caso le enviará su punto de vista.

La figura 4.8 muestra la representación AUML del protocolo descrito.

Respecto de la representación AUML del protocolo de Sian para dos agentes, puede apreciarse que en esta ocasión el protocolo es simétrico para todos los agentes, por lo que todos ellos utilizarán la misma especificación ACSL.

Se muestra a continuación la especificación en ACSL del último hilo de interacción correspondiente al rol iniciador de la figura 4.8. En ella puede apreciarse la declaración de los parámetros (construcción *paramSetDelc*, la ulterior referencia a éstos parámetros (construcción *paramSet Ref*) y la instanciación de los parámetros de otros hilos de interacción referenciados con los valores de éstos (construcción *paramSetInst*).

```
<threadOfInteraction threadName='WaitAnswer'>
```

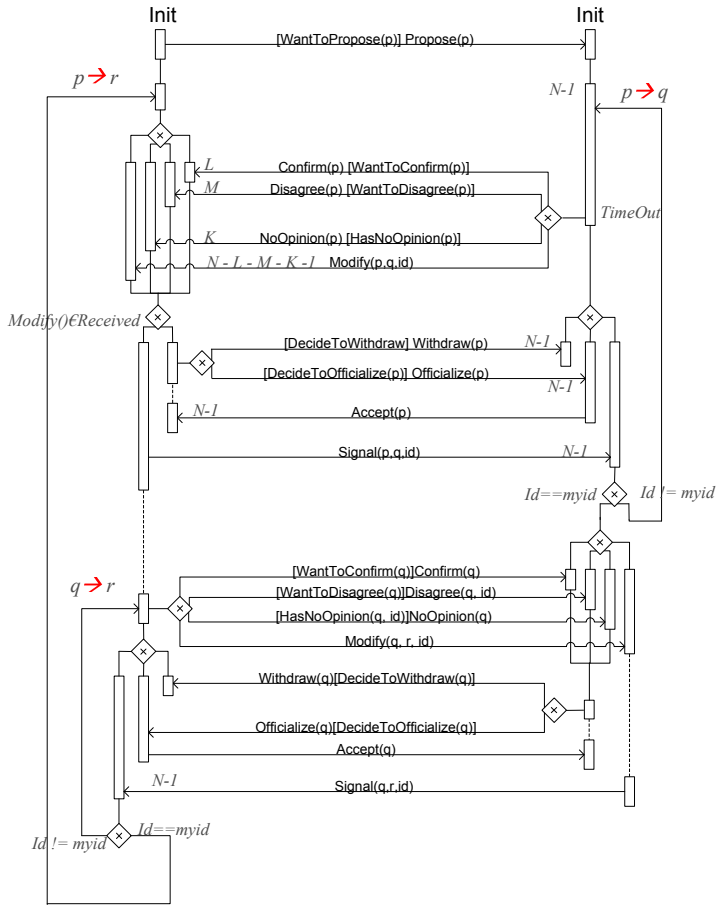

Figura 4.8: Especificación AUML del protocolo de coordinación de Sian para más de dos agentes

```
<paramSetDecl name='decisionProposeParams'>
 <param>
  <name>p</name>
   <type>urn:TIC2001-3451:schemas:acsl:parameterTypes:proposition</type>
 </param>
 <param>
  <name>q</name>
  <type>urn:TIC2001-3451:schemas:acsl:parameterTypes:proposition</type>
 </param>
 <param>
  <name>id</name>
  <type>urn:TIC2001-3451:schemas:acsl:parameterTypes:agentID</type>
 </param>
 <param>
  <name>newid</name>
  <type>urn:TIC2001-3451:schemas:acsl:parameterTypes:agentID</type>
 </param>
</paramSetDecl>
<pick>
   ...
 <eventHandler>
```

```
   <event>
    <exchange message='Signal' direction='in' mode='middle'>
     <paramSetRef>
      <paramRef>p</paramRef>
      <paramRef>q</paramRef>
      <paramRef>newid</paramRef>
     </paramSetRef>
    </exchange>
   </event>
   <action>
    <switch multiChoice='false'>
     <branch>
      <case condition='equals(id,newid)'>
       <paramSetRef>
        <paramRef>id</paramRef>
        <paramRef>newid</paramRef>
       </paramSetRef>
      </case>
      <action>
       <threadOfInteractionRef threadRef='OpinionPropose'>
        <paramInstSet>
         <paramInst>
          <ref>p</ref>
          <value>q</value>
         </paramInst>
        </paramInstSet>
       </threadOfInteractionRef>
      </action>
     </branch>
     <default>
      <action>
       <threadOfInteractionRef threadRef='OpinionAnswer'>
        <paramInstSet>
         <paramInst>
          <ref>p</ref>
          <value>q</value>
         </paramInst>
        </paramInstSet>
       </threadOfInteractionRef>
      </action>
     </default>
    </switch>
   </action>
  </eventHandler>
 </pick>
</threadOfInteraction>
```

A continuación se recogen las reglas del intérprete correspondiente al agente que inicia la conversación:

$$\langle\langle \text{Init}, p_1 \in B_e\rangle, WantToPropose(p_1)\rangle \xrightarrow{Propose(p_1)} \langle\langle \text{Opinion}, p_1\rangle, NOOP\rangle$$

$$\langle\langle \text{Init}, p_1 \in B_e\rangle, WantToAssert(p_1)\rangle \xrightarrow{Assert(p_1)} \langle\langle \text{AgreementPropose}, p_1\rangle, NOOP\rangle$$

$$\langle\langle \text{AgreementPropose}, p_1\rangle, true\rangle \xrightarrow[Accept(p_1)]{} \langle\langle \text{End}, p_1\rangle, B_e - p_1, B_c + p_1\rangle$$

$$\langle\langle \text{AgreementPropose}, p_1\rangle, timeout\rangle \overrightarrow{\varepsilon} \langle\langle \text{End}, p_1\rangle, \text{NOOP}\rangle$$

$\langle\langle \text{Opinion}, p_1\rangle, true\rangle \xrightarrow[p_1, Confirm(p_1)]{} \langle\langle \text{DecisionPropose}, Confirm(p_1)\rangle, NOOP\rangle$

$\langle\langle \text{Opinion}, p_1\rangle, true\rangle \xrightarrow[p_1, Disagree(p_1)]{} \langle\langle \text{DecisionPropose}, Disagree(p_1)\rangle, NOOP\rangle$

$\langle\langle \text{Opinion}, p_1\rangle, true\rangle \xrightarrow[p_1, NoOpinion(p_1)]{} \langle\langle \text{DecisionPropose}, NoOpinion(p_1)\rangle, NOOP\rangle$

$\langle\langle \text{Opinion}, p_1\rangle, true\rangle \xrightarrow[Modify(p_1, p_2)]{} \langle\langle \text{CounterOpinion}, p_2\rangle, B_e - p_1, B_c\rangle$

$\langle\langle \text{Opinion}, p_1\rangle, timeout\rangle \xrightarrow{\varepsilon} \langle\langle \text{End}, p_1\rangle, \text{NOOP}\rangle$

$\langle\langle \text{DecisionPropose}, p_1, s(p_1)\rangle, DecideToWithdraw(p_1, s(p_1))\rangle \xrightarrow{Withdraw(p_1)} \langle\langle \text{End}, p_1\rangle, B_e - p_1, B_c\rangle$

$\langle\langle \text{DecisionPropose}, p_1, s(p_1)\rangle, DecideToOfficialize(p_1, s(p_1))\rangle \xrightarrow{Officialize(p_1)} \langle\langle \text{AgreementPropose}, p_1\rangle, NOOP\rangle$

$\langle\langle \text{DecisionAnswer}, p_1\rangle, true\rangle \xrightarrow[Withdraw(p_1)]{} \langle\langle \text{End}, p_1\rangle, B_e - p_1, B_c\rangle$

$\langle\langle \text{DecisionAnswer}, p_1\rangle, true\rangle \xrightarrow[Officialize(p_1)]{} \langle\langle \text{AgreementAnswer}, p_1\rangle, NOOP\rangle$

$\langle\langle \text{DecisionAnswer}, p_1\rangle, timeout\rangle \xrightarrow{\varepsilon} \langle\langle \text{End}, p_1\rangle, \text{NOOP}\rangle$

$\langle\langle \text{AgreementAnswer}, p_1\rangle, true\rangle \xrightarrow{Accept(p_1)} \langle\langle \text{End}, p_1\rangle, B_e - p_1, B_c + p_1\rangle$

$\langle\langle \text{CounterOpinion}, p_1\rangle, WantToConfirm(p_1)\rangle \xrightarrow{Confirm(p_1)} \langle\langle \text{DecisionAnswer}, p_1\rangle, NOOP\rangle$

$\langle\langle \text{CounterOpinion}, p_1\rangle, WantToDisagree(p_1)\rangle \xrightarrow{Disagree(p_1)} \langle\langle \text{DecisionAnswer}, p_1\rangle, NOOP\rangle$

$\langle\langle \text{CounterOpinion}, p_1\rangle, NoOpinion(p_1)\rangle \xrightarrow{NoOpinion(p_1)} \langle\langle \text{DecisionAnswer}, p_1\rangle, NOOP\rangle$

$\langle\langle \text{CounterOpinion}, p_1\rangle, WantToModify(p_1, p_2)\rangle \xrightarrow{Modify(p_1, p_2)} \langle\langle \text{Opinion}, p_2\rangle, B_e - p, B_c\rangle$

Las siguientes reglas corresponden al intérprete para el agente que recibe la propuesta inicial:

$\langle\langle \text{Init}, p_1 \in B_e\rangle, true\rangle \xrightarrow[Propose(p_1)]{} \langle\langle \text{Opinion}, p_1\rangle, NOOP\rangle$

$\langle\langle \text{Init}, p_1 \in B_e\rangle, true\rangle \xrightarrow[Assert(p_1)]{} \langle\langle \text{AgreementPropose}, p_1\rangle, NOOP\rangle$

$\langle\langle \text{AgreementPropose}, p_1\rangle, true\rangle \xrightarrow{Accept(p_1)} \langle\langle \text{End}, p_1\rangle, B_e - p_1, B_c + p_1\rangle$

$\langle\langle \text{Opinion}, p_1\rangle, WantToConfirm(p_1)\rangle \xrightarrow{Confirm(p_1)} \langle\langle \text{DecisionPropose}, p_1\rangle, NOOP\rangle$

$\langle\langle \text{Opinion}, p_1\rangle, WantToDisagree(p_1)\rangle \xrightarrow{Disagree(p_1)} \langle\langle \text{DecisionPropose}, p_1\rangle, NOOP\rangle$

$\langle\langle \text{Opinion}, p_1\rangle, HaveNoOpinion(p_1)\rangle \xrightarrow{NoOpinion(p_1)} \langle\langle \text{DecisionPropose}, p_1\rangle, NOOP\rangle$

$\langle\langle \text{Opinion}, p_1\rangle, WantToModify(p_1, p_2)\rangle \xrightarrow{Modify(p_1, p_2)} \langle\langle \text{CounterOpinion}, p_2\rangle, B_e - p_1, B_c\rangle$

$\langle\langle \text{CounterOpinion}, p_1\rangle, true\rangle \xrightarrow[Confirm(p_1)]{} \langle\langle \text{DecisionAnswer}, Confirm(p_1)\rangle, NOOP\rangle$

$\langle\langle \text{CounterOpinion}, p_1\rangle, true\rangle \xrightarrow[Disagree(p_1)]{} \langle\langle \text{DecisionAnswer}, Disagree(p_1)\rangle, NOOP\rangle$

$\langle\langle \text{CounterOpinion}, p_1\rangle, true\rangle \xrightarrow[NoOpinion(p_1)]{} \langle\langle \text{DecisionAnswer}, NoOpinion(p_1)\rangle, NOOP\rangle$

$\langle\langle \text{CounterOpinion}, p_1\rangle, true\rangle \xrightarrow[Modify(p_1, p_2)]{} \langle\langle \text{Opinion}, p_2\rangle, B_e - p, B_c\rangle$

$\langle\langle \text{CounterOpinion}, p_1\rangle, timeout\rangle \xrightarrow{\varepsilon} \langle\langle \text{End}, p_1\rangle, \text{NOOP}\rangle$

$\langle\langle \text{DecisionPropose}, p_1\rangle, true\rangle \xrightarrow[Withdraw(p_1)]{} \langle\langle \text{End}, p_1\rangle, B_e - p_1, B_c\rangle$

$\langle\langle \text{DecisionPropose}, p_1\rangle, true\rangle \xrightarrow[Officialize(p_1)]{} \langle\langle \text{AgreementPropose}, p_1\rangle, NOOP\rangle$

$\langle\langle \text{DecisionPropose}, p_1\rangle, timeout\rangle \xrightarrow{\varepsilon} \langle\langle \text{End}, p_1\rangle, \text{NOOP}\rangle$

$\langle\langle \text{AgreementAnswer}, p_1\rangle, true\rangle \xrightarrow[Accept(p_1)]{} \langle\langle \text{End}, p_1\rangle, B_e - p_1, B_c + p_1\rangle$

$\langle\langle \text{AgreementAnswer}, p_1\rangle, timeout\rangle \xrightarrow{\varepsilon} \langle\langle \text{End}, p_1\rangle, \text{NOOP}\rangle$

$\langle\langle \text{DecisionAnswer}, p_1, s(p_1)\rangle, DecideToWithdraw(p_1, s(p_1))\rangle \xrightarrow{Withdraw(p_1)} \langle\langle \text{End}, p_1\rangle, B_e - p_1, B_c\rangle$

$\langle\langle \text{DecisionAnswer}, p_1, s(p_1)\rangle, DecideToOfficialize(p_1, s(p_1))\rangle \xrightarrow{Officialize(p_1)} \langle\langle \text{AgreementAnswer}, p_1\rangle, NOOP\rangle$

Capítulo 5

Modelo de Organización

Índice General

5.1. Aspectos organizativos de la gestión 140

5.2. Necesidad de un nuevo modelo de organización de tipo cooperativo 142

5.3. Organizaciones holónicas 144

5.4. Modelo de Organización Holónico 145

5.5. Caso de aplicación: Gestión holónica de los aspectos de seguridad de un SLA 154

5.6. Políticas sociales 158

5.7. Dinámica de las obligaciones 166

5.8. Creación, derogación y distribución de políticas mediante actos ilocucionales 168

El Modelo de Organización de una arquitectura de gestión define las relaciones existentes entre los sistemas de gestión y los recursos gestionados. Con este propósito define los actores, sus roles e interrelaciones, sus posibilidades de agrupación en dominios o vistas y los principios fundamentales para su cooperación. Si el modelo provee del concepto de dominio o vista, debe especificar cómo se forman los grupos, cómo se asignan responsabilidades y reglas de gestión a éstos y cómo pueden modificarse estas asignaciones dinámicamente. La asignación de responsabilidades requerirá de un modelo de delegación que es parte del modelo de organización y que contempla la descripción, el transporte y la efectividad de los scripts de gestión. Para la especificación de las reglas de gestión específicas de un dominio se suele utilizar el concepto de política. Las políticas definen reglas de gestión con un alto nivel de abstracción, derivadas de los objetivos corporativos o los procesos de TI. Si el modelo organizativo contempla el concepto de política, deberá contemplar su especificación, organización, distribución e interpretación/traducción.

El modelo no debe dictar las estructuras organizativas y operacionales de los operadores de los sistemas ni de los proveedores de los servicios a gestionar. En su lugar, debería ajustarse a la gran variedad de formas que caracterizan a dichas organizaciones proporcionando opciones adecuadas que faciliten su adaptación a estas formas. Esto significa que la arquitectura debe facilitar una descripción de formas de cooperación, de roles, del modo en que se organizan grupos, etc.

Los sistemas de gestión pueden adoptar diferentes organizaciones topológicas y funcionales. En la *gestión centralizada*, un único sistema de gestión monopoliza la responsabilidad para todas las tareas. En la *gestión multipunto*, los recursos se combinan en grupos (dominios) en función de diferentes aspectos (topológicos, funcionales, organizativos), y se asigna un gestor a cada grupo. Si se desea coordinar los diferentes gestores desde una perspectiva de control multipunto, entonces, dependiendo del esquema de cooperación, esto produce (a) un control multicentro, (b) una disposición jerárquica (multinivel), en la que los gestores intermedios preprocesan datos, analizan umbrales y filtran o correlan sucesos, o (c) una disposición heterárquica y cooperativa como la propuesta en esta tesis, en la que los diferentes gestores forman una red de cooperación para llevar a cabo la gestión. Se pasa así de una cooperación asimétrica típica del modelo cliente-servidor adoptado por las arquitecturas jerárquicas a una cooperación simétrica. El paso intermedio entre estas dos formas de cooperación lo suponen las arquitecturas de gestión basadas en el modelo igual-a-igual de infraestructuras de objetos distribuidos como CORBA. La diferencia fundamental es la granularidad de la delegación (operaciones frente a metas), la complejidad de los procesos de cooperación (invocación de operaciones y paso de información en ambos sentidos frente a esquemas de negociación soportados por protocolos de interacción) y el grado de autonomía de las partes cooperantes (invocaciones de métodos frente a actos hablados).

5.1. Aspectos organizativos de la gestión

La aproximación integrada a la gestión distribuida persigue principalmente la elaboración de soluciones afines a la pirámide de gestión implícita en la recomendación ITU-T M.3010. Esta pirámide estratifica la funcionalidad y los objetivos de una solución de gestión en: gestión de elementos, gestión de redes, gestión de sistemas (distribuidos), gestión de información, gestión de aplicaciones y servicios, gestión de clientes y gestión de negocio. Pero además, una aproximación integrada a la gestión exigirá adaptar las soluciones a la estructura organizativa y operativa de la empresa objetivo. Este último hecho condiciona en gran medida el proceso de elaboración del Modelo de Organización de la arquitectura de gestión. Con el fin de contemplar los

aspectos organizativos y operativos de las empresas, dicho proceso deberá considerar las siguientes actividades[1]:

- Definición de las políticas y los procesos de gestión que soportan los diferentes procesos de negocio.
- Definición de los roles implicados en dichos procesos.
- Definición de los dominios a los que afectan las políticas y los procedimientos de gestión asociados a dichos procesos. El concepto de dominio extiende al de rol y permite definir la estructura organizativa (y operativa) de la solución de gestión. De entre los posibles criterios existentes para el establecimiento de dominios, destacan los basados en (1) la estructura organizativa de la compañía (equipos, grupos, departamentos, áreas operativas), (2) aspectos geográficos (país, ciudad, edificio, planta), (3) áreas de negocio, (4) aspectos funcionales, (5) tipos de recursos (equipos, redes), (6) políticas aplicables, (7) necesidades de cooperación, etc.
- Especificación de las interfaces de cooperación existentes entre dominios, a través de las cuales se producirá el intercambio de información y la actividad de gestión (procesos de negociación, delegación de tareas, solicitud de servicio, etc.).
- Especificación y distribución/asignación de información, servicios, tareas y metas (responsabilidades) a (y entre) dominios (unidades organizativas). El modo en que se lleve a cabo esta distribución de responsabilidades entre los diferentes dominios resultará crucial a la hora de determinar los requisitos de cooperación entre dichos dominios y marcará un punto de inflexión determinante en la evolución de los modelos organizativos centralizados y jerárquico-distribuidos a un modelo cooperativo.

Hasta hace poco tiempo, los modelos de organización que mejor se adaptaban a la estructura organizativa y operativa de las empresas eran los modelos centralizados y los modelos jerárquicos. Sin embargo, con la liberalización del mercado de las telecomunicaciones y la aparición y proliferación del concepto de Empresa Virtual en el mundo de los servicios telemáticos, los requisitos de gestión han cambiado, precisando cada vez más la evolución hacia modelos de organización heterárquicos

[1]Se enumeran tan solo las actividades relacionadas con la elaboración del modelo organizativo de la arquitectura de gestión. Otros autores como [HAN98] prestan más atención a las actividades relacionadas con la organización de la infraestructura de IT (humana y material).

o cooperativos. Una *Empresa Virtual* (EV) es una red flexible de cooperación temporal guiada por las metas, compuesta por compañías independientes que se reúnen rápidamente y contribuyen con sus capacidades/aptitudes con el fin de explotar una oportunidad de mercado específica (y por tanto, con el fin de proveer un servicio demandado o producto según un entendimiento común del negocio). Aunque las EV rechazan todo tipo de institucionalización, y no disponen de oficina central (*central office*), ni de jerarquía organizativa, ni de integración vertical, las compañías autónomas asociadas actúan hacia el exterior como una única corporación. Debido a que su cooperación se lleva a cabo a través de tecnologías de información y de comunicación on-the-edge, su estructura organizativa es fluida y flexible y evoluciona a lo largo del tiempo [RV]. Una EV se forma por tanto para llevar a cabo una o más metas específicas. Una vez cumplida(s) la meta(s), la EV puede diseminarse o evolucionar cambiando su meta. La investigación en EV está orientada predominantemente a la utilización de tecnologías *GroupWare* y CSCW (*Computer Supported Cooperative Work*).

5.2. Necesidad de un nuevo modelo de organización de tipo cooperativo

Los sistemas distribuidos y los nuevos servicios emergentes de telecomunicaciones comparten un conjunto de características en torno a las cuales se origina la demanda de un nuevo modelo de organización más flexible de tipo cooperativo. Entre estas características podemos destacar:

- Dinamicidad. Los holones facilitan la reestructuración organizativa en función de la meta.
- Distribución geográfica.
- Ausencia de una autoridad de control central y de una estructura organizativa de tipo jerárquica. Debido a la liberalización del mercado de las comunicaciones y de los servicios telemáticos entran en juego numerosas entidades autónomas con intereses enfrentados. Deja de tener sentido un liderazgo basado en el control sobre qué hacer y sobre los resultados, para dar paso a un liderazgo basado en la coordinación y en la creación de un marco de trabajo para participación colaborativa.

  La ausencia de control central requiere de una política flexible de control de acceso, obligaciones, etc. Que tenga en cuenta las relaciones existentes entre los holones y entre los roles de sus miembros.

- Demanda de cooperación basada en las tecnologías de información y comunicaciones. La ausencia de una autoridad de control central implica la necesidad de disponer de tecnologías de cooperación en lugar de información que fluye por una jerarquía.

- Diferentes culturas organizativas. Junto con el cambio o incluso la disolución de la estructura organizativa, la cultura organizativa se hace cada vez más importante. La diferencia entre culturas organizativas provoca que la confianza mutua se haga cada vez más importante en los procesos de cooperación pero también cada vez más difícil de establecer. El uso de estructuras normativas facilitará esta tarea.

- Conciencia, vista como el entendimiento de las actividades de los demás, el cual proporciona un contexto para la propia actividad. Relacionado con reputación, monitorización, etc.

- Balance entre conciencia y privacidad mediante el establecimiento de parámetros SLA. Debe existir una política que explicite qué información y qué servicios están accesibles. Esta política se define a nivel de dominio de cooperación. Las aproximaciones basadas en autorizaciones de acceso de tipo jerárquico gestionadas por una autoridad central dejan de ser válidas. Ahora se requiere mayor distribución y dinamismo en su gestión, que vendrán dados por las políticas de delegación. Estas políticas deben contemplar la posibilidad de negociar los derechos. Debe contemplarse también un mecanismo de resolución de conflictos entre políticas (e.g. basado en atribuir diferentes pesos/prioridades a las diferentes políticas, basado en sus especificidades (mayor especificidad implica mayor prioridad), basado en procesos de deliberación acerca del predicado violación, etc.).

5.2.1. Requisitos del modelo de organización en el paradigma cooperativo

A la vista de estas necesidades, puede decirse que el modelo de organización de una arquitectura de gestión concebida bajo el paradigma cooperativo debe permitir el logro de las siguientes características:

Autonomía: Cada unidad organizativa debe ser capaz de crear, controlar y monitorizar la ejecución de sus propios planes y/o estrategias de gestión, y realizar acciones correctivas/preventivas adecuadas (racionales) contra sus propias disfunciones.

Cooperación: Las diferentes unidades organizativas deben ser capaces de negociar y ejecutar planes mutuamente aceptables (joint intentions) y llevar a cabo acciones correctivas/preventivas contra disfunciones globales.

Carácter abierto: El sistema debe permitir la inclusión de nuevas unidades organizativas, la eliminación de las existentes y la modificación de las capacidades funcionales de las existentes, con intervención humana mínima. Las unidades organizativas o sus funciones podrían ser proporcionados por diferentes fuentes (heterogeneidad).

5.3. Organizaciones holónicas

El ganador del premio Nobel de Ciencias Económicas e investigador en el campo de la inteligencia artificial Herbert. A. Simon [Sim90], a partir de la interpretación de la "parable of the two matchmakers", concluyó que los sistemas evolucionan mucho más rápidamente de un nivel de complejidad determinado a niveles de complejidad significativamente superiores si se dispone de formas intermedias estables que actúen como pasos intermedios hacia el estado objetivo. Retomando esta conclusión, el filósofo húngaro Arthur Koestler [Koe68] introdujo el concepto de holón, término que se compone de la palabra griega "holos", que significa "el todo" y el sufijo "on" que evoca una partícula o "parte". El concepto está inspirado en la observación de las estructuras recursivas autosimilares presentes en los sistemas organizativos biológicos. Hace 25 años, su inspirador, definió un holón como una parte identificable de un sistema que tiene identidad única, compuesta de partes subordinadas y que, a su vez, es parte de un todo mayor.

El término holón se emplea para denominar entidades que exhiben simultáneamente un comportamiento autónomo (se comportan como un todo) y no autosuficiente (se comportan como una parte de un todo mayor), por lo que requieren capacidades de cooperación. Se puede definir entonces un holón como un bloque de construcción con entidad propia, que presenta un comportamiento autónomo y cooperativo. Esta dualidad demanda un equilibrio entre las fuerzas contradictorias que definen cada una de estas propiedades a nivel de comportamiento. Por autonomía se entiende aquí la capacidad de una entidad para crear sus propios planes y/o estrategias y controlar su ejecución, así como su propio estado. Por cooperación se entiende el proceso por el que un conjunto de entidades desarrollan planes comúnmente aceptados y los llevan a cabo de forma distribuida.

Una de las principales características de un holón es su granularidad múltiple, que se manifiesta a través de replicación en estructuras autosimilares. Esta des-

composición jerárquica/heterárquica resulta en una jerarquía anidada de entidades "fractales" denominada holarquía. Una holarquía puede definirse pues como un sistema de holones autorregulados que cooperan para alcanzar una meta u objetivo global. La holarquía define las reglas básicas para la cooperación de los holones que la forman y por tanto limita su autonomía y/o influye en sus procesos de toma de decisiones.

Se denomina organización holónica a una holarquía de organizaciones colaborativas consideradas como holones. El modelo holónico de organización combina las mejores características de sus homólogos dominantes: (1) el modelo jerárquico o top-down y (2) el modelo heterárquico, cooperativo o bottom-up. Las propiedades asociadas tradicionalmente a un holón, y a las organizaciones holónicas son las de ser eficientes en el uso de recursos, muy resistentes y estables ante perturbaciones (tanto internas como externas), adaptables a los cambios en el entorno en que existen y escalables a sistemas complejos. Su fortaleza y estabilidad procede de ser unidades independientes en el tratamiento de problemas cooperativos en entornos heterogéneos. Su eficiencia (y la del sistema global) proviene del hecho de que sean a su vez unidades subordinadas de otros (múltiples) holones de nivel superior que les instruyen y, en cierta medida, controlan.

El carácter cooperativo de los holones se observa en los procesos de planificación y asignación que se realizan para la consecución las metas globales del sistema en que se integran, así como en la resolución de sus de carencias (en recursos y/o capacidades) para la resolución de sus propias metas. Su carácter autónomo se observa en la consecución de sus metas propias, así como en la realización de las tareas asignadas en pro de la consecución de las metas globales.

5.4. Modelo de Organización Holónico

Se introduce el término organización para designar una agrupación formal y estable de agentes que ofrece una interfaz bien definida de comunicación con el exterior y que constituye en si misma una unidad independiente e identificable (con entidad propia). Si se equipara ésta interfaz a la ofrecida por un agente (y se le aplican relaciones y estructuras normativas semejantes, y por tanto responsabilidades, obligaciones, etc.), por el principio de encapsulación podremos concebir y manejar dicha organización como si se tratase de un único agente, abstrayéndonos de su composición interna. Si se admite recursividad en esta definición (y por tanto una organización puede agrupar otras organizaciones) y se le atribuyen a los elementos de su estructura interna las características deseables de autonomía, cooperación y carácter abierto, estaremos entonces frente a un modelo organizativo de tipo holó-

nico.

Una organización con estructura holónica está por tanto compuesta por subelementos autónomos y cooperativos denominados unidades organizativas u holones que se organizan en una jerarquía/heterarquía de contención denominada holarquía. Las actividades de cada unidad organizativa vienen determinadas por procesos de cooperación con otras unidades, y no por un mecanismo centralizado. Este último resulta difícilmente escalable, e incluso impracticable, en el caso de unidades organizativas heterogéneas y/o con intereses propios. En el modelo organizativo de tipo holónico que se propone se diferencia entre agente atómico y agente holónico u holón. Un holón representa así una organización (unidad organizativa) que agrupa un conjunto de agentes (a su vez atómicos u holónicos) pero interactúa con otros agentes externos como un único agente atómico. Este proceso de agrupación es recursivo, ya que el modelo holónico permite de manera explícita que las organizaciones sean miembros de otras organizaciones para configurar una holarquía. Esta aproximación holónica permite ver a las organizaciones como entidades autónomas, comunicativas y cooperativas.

La figura 5.1 muestra la composición de una holarquía.

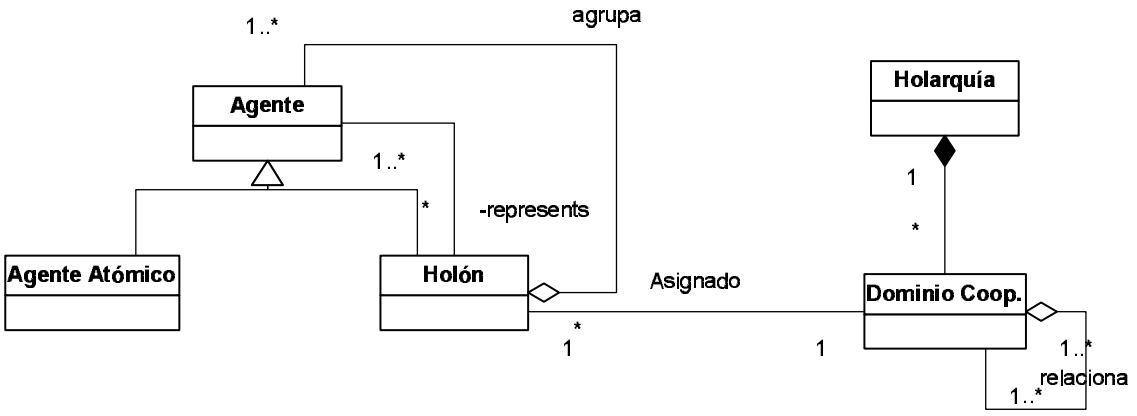

Figura 5.1: Composición de una holarquía

Así, las organizaciones holónicas son simultáneamente tanto totalidades como partes. Esto quiere decir que cualquier organización es simultáneamente (a) una parte de un todo mayor (una unidad organizativa de una organización superior o una sociedad), y (b) una unidad relativamente independiente. Como holón, una organización puede por tanto contemplarse tanto como una unidad individual aislada, como formando parte de un contexto colectivo.

Los agentes autónomos pueden asociarse y formar holones. Los agentes de un holón deben perseguir al menos una meta común y por tanto muestran un compor-

tamiento cooperativo dirigido por dicha meta. Aun así, continúan siendo libres para reconfigurar el holón (cambio de meta) e incluso abandonarlo y actuar de forma autónoma o formar nuevos holones con otros agentes.

La cooperación entre los componentes de un holón se lleva a cabo a través de los así denominados dominios de cooperación.

5.4.1. Dominios de cooperación

Un dominio de cooperación se define como un espacio lógico en el que los holones operan y se comunican, proporcionando con ello el contexto en el que estos holones pueden localizarse, contactar e interactuar unos con otros. Es posible que un dominio de cooperación no exista por si mismo, y que todos ellos se generen dinámicamente por las operaciones de los propios holones. Las siguientes premisas son válidas para un dominio de cooperación: Todo sistema de gestión holónico contiene al menos un dominio de cooperación (las formas más simples son los sistemas centralizados y los jerárquicos débilmente distribuidos). Un holón puede ser miembro de uno o más dominios de cooperación (posiblemente anidados). Un dominio de cooperación tiene uno o más holones como miembros.

La figura 5.2 extiende el modelo holárquico presentado en la figura 5.1 para incorporar los conceptos de rol, puesto de trabajo y dominio de cooperación. En el contexto de esta figura y con el fin de no establecer relaciones entre conceptos con diferentes niveles de abstracción, la relación existente entre los conceptos *Agente* y *Holón* se entiende en términos del agente que actúa como cabeza del holón.

El apéndice D recoge la especificación OWL de este modelo, así como su especificación en lógica descriptiva desde dos aproximaciones diferentes. La primera aproximación contempla el modelo presentado como tal, permitiendo por tanto su instanciación directa en una base de conocimiento asertivo ABox (también recogida en el mismo apéndice como solución del caso práctico propuesto en la sección 5.5). La segunda aproximación contempla realmente el modelo presentado como un metamodelo, permitiendo con ello su instanciación en diferentes modelos. Esta solución resulta más flexible a la hora de modelar organizaciones de gran tamaño organizadas en torno a unidades estructurales que presentan configuraciones similares, si no idénticas, en términos de roles y relaciones entre éstos. Así, cada modelo, instancia del metamodelo recogido en la figura 5.2, puede instanciarse sucesivas veces para representar cada una de estas unidades funcionales. La sección 5.5 presenta un caso de aplicación que refleja esta necesidad. Este caso muestra además como esta última aproximación permite mayor expresividad a la hora de, por ejemplo, establecer el carácter exclusivo (unicidad) de un rol (un rol único en una unidad organizativa no

tiene por qué ser único en todas las instanciaciones de dicha unidad).

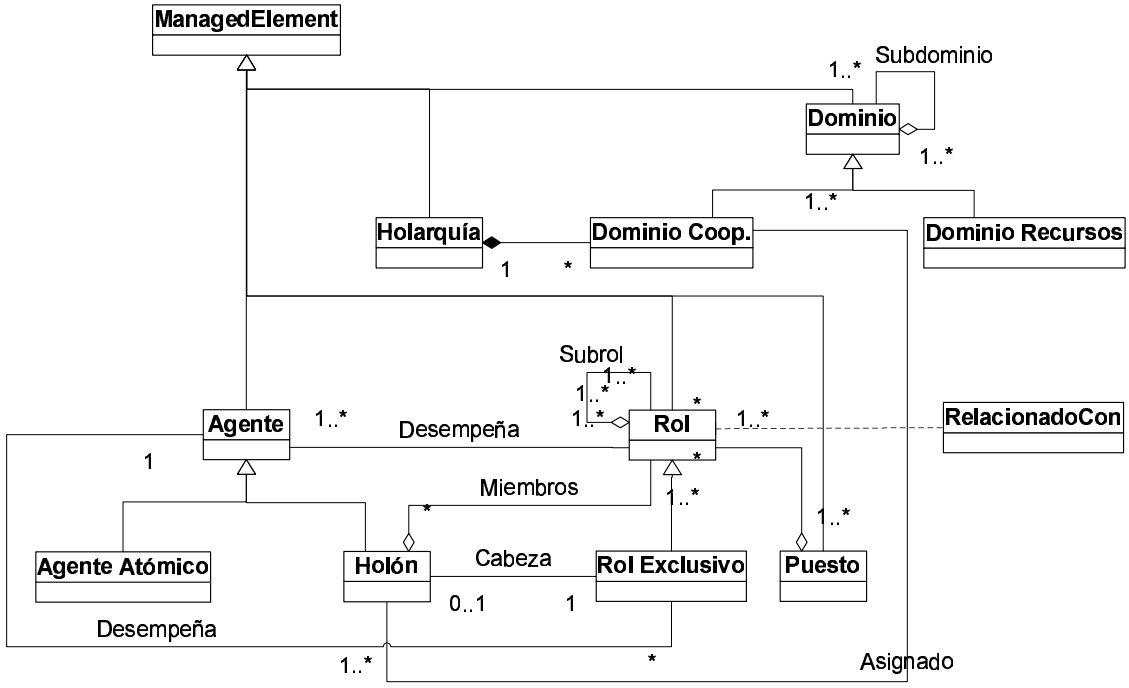

Figura 5.2: Composición de una holarquía orientada a roles

El beneficio de contemplar un agente atómico como un holón y dotarle de dominio de cooperación (verlo como un "head") incrementa la flexibilidad del sistema al poder ser intercambiado fácilmente por un holón compuesto sin que éste cambio afecte al sistema.

Una organización holónica puede llevar a cabo estrategias de (1) negociación en el acuerdo de tareas, (2) planificación coordinada durante la planificación de tareas, (3) resolución de conflictos durante la ejecución de las tareas y (4) compensación de transacciones (o *compromiso en dos fases* cuando sea posible) para asegurar la consistencia de la información. Cada estrategia requiere el intercambio de conocimiento y la solicitud de actividades a través de un único dominio de cooperación asociado a la tarea.

Desde esta perspectiva, el concepto de *holón* propuesto por Arthur Koestler guarda estrecha relación con el concepto de *holismo* introducido previamente por Aristóteles [Ari73] para expresar cómo *el todo* siempre es más que la suma de *las partes*. Idea que luego han utilizado algunos autores destacados tanto en el área de la Ingeniería del Software como en el campo de la Sociología. Así, el sociólogo positivista francés Èmile Durkheim utilizó el término para describir el concepto de *holismo sociológico* o *sociologismo determinista* [Dur94] en el que los grupos sociales y organizativos son irreductibles, ya que no suponen simplemente la suma de los

individuos que los componen, sino que forman un *ente superior*; se trata entonces de la sociedad como un ente trascendente que actúa como un todo a la hora de deteminar las maneras de hacer y las maneras de ser colectivas de los individuos que la componen. Del mismo modo, Grady Booch, desde la visión de modelado de objetos, promulgó en [Boo94] que una agregación es más que la suma de las partes agregadas.

En este mismo sentido, el modelo de organización holónica propuesto en este trabajo de Tesis da solución también al problema clásico de la Ingeniería del Software que se plantea al subdividir un problema en subproblemas. Esta división origina elementos completamente autocontenidos, y elementos que pueden originar efectos colaterales no deseados en otros elementos. La utilización del modelo propuesto eliminaría, o al menos limitaría, los conflictos ocasionados por la ejecución de estos últimos, al constituirlos como elementos holónicos y concentrar su interrelación en los conocimientos y en la funcionalidad de los agentes que actúan como *cabezas* de los holones correspondientes[2].

La cooperación entre holones ocurre siempre a través de sus respectivos dominios de cooperación. Un dominio de cooperación comprende los siguientes elementos:

1. Estructuras de información y ontologías compartidas por sus holones miembros.

2. Interfaces de cooperación basadas en protocolos de interacción.

3. Infraestructura para el paso de mensajes entre las organizaciones y el dominio de cooperación. Si el dominio de cooperación se localiza en el mismo dispositivo físico, el paso de mensajes puede conseguirse por memoria compartida. En otro caso, los mensajes deben representarse, codificarse y enviarse a través de una red de comunicaciones. Este aspecto se aborda en el modelo de comunicación.

4. Mecanismos de toma de decisiones que ayuden a los holones en sus actividades de planificación de tareas, negociación, intercambio de información, etc.

5. Técnicas y reglas de descomposición y asignación de tareas entre los miembros del dominio, así como facilidades para la planificación y el control de tareas dentro de un holón (síntesis).

6. Facilidades para la monitorización del estado de una tarea distribuida, y la planificación y control de todas las acciones que conforman la tarea.

[2]Esta última idea surgió a raíz de una discusión mantenida sobre el tema por el autor con el Prof. Dr. Juan Pazos, Catedrático de Universidad del Depto. de Inteligencia Artificial de la Universidad Politécnica de Madrid

7. Facilidades de pasarela basada en políticas que restrinjan, controlen y monitorizen la comunicación que esté teniendo lugar.

En una organización holónica, las unidades organizativas (holones) de nivel inferior incluidos dentro de la misma cooperan entre sí a través de sus respectivos dominios de cooperación para generar planes y para llevar éstos a cabo. En el caso de que un holón no incluya holones de nivel inferior (y por tanto sea un agente atómico), el dominio de cooperación interno representa la arquitectura interna del agente. La estructura arquitectónica del dominio de cooperación descrito aquí se fundamenta en las propiedades y el comportamiento de los componentes (holones) que contribuyen al dominio de cooperación.

Dentro de un dominio de cooperación se consideran dos tipos de cooperación entre holones:

Cooperación simple: Un sistema autónomo se compromete a conversar con otro sistema siguiendo un conjunto de protocolos de interacción preestablecidos. Se consideran aceptables respuestas no cooperativas (refuse, not-understood, etc.). Todos los holones deben presentar facilidades de cooperación simple.

Cooperación compleja: Encaminada a alcanzar una meta compartida (joint goal) por varios holones, por ejemplo el acuerdo de un plan común para ejecutar las tareas asociadas a un problema distribuido.

La estructura creada por esta holarquía de tareas es dinámica, mientras que las relaciones entre holones forman una configuración más estable. Los holones responden a solicitudes de tareas provenientes de sus dominios de cooperación, de modo que la interacción es realizada o se generan nuevas tareas de acuerdo con estas respuestas. Si una tarea no puede ejecutarse debido a la falta de recursos o capacidades, entonces puede alterarse o introducirse un nuevo componente en el holón para satisfacer los requisitos del dominio (delegación dinámica). Como resultado, la estructura interna de la holarquía representada por este dominio de cooperación se altera a través de la generación de un nuevo holón compuesto.

Cada holón entra a formar parte de un dominio de cooperación interactuando con un interfaz de dominio de cooperación (subscribe & publish) de modo que pueda adquirir/presentar información de/a cada dominio de cooperación.

5.4.1.1. Expresiones de Dominio (EdD) y lógica descriptiva

Con el fin de permitir combinar dominios y expresar el conjunto de roles a los que se aplica una política se introduce un lenguaje declarativo ℓ_{dom} para construcción

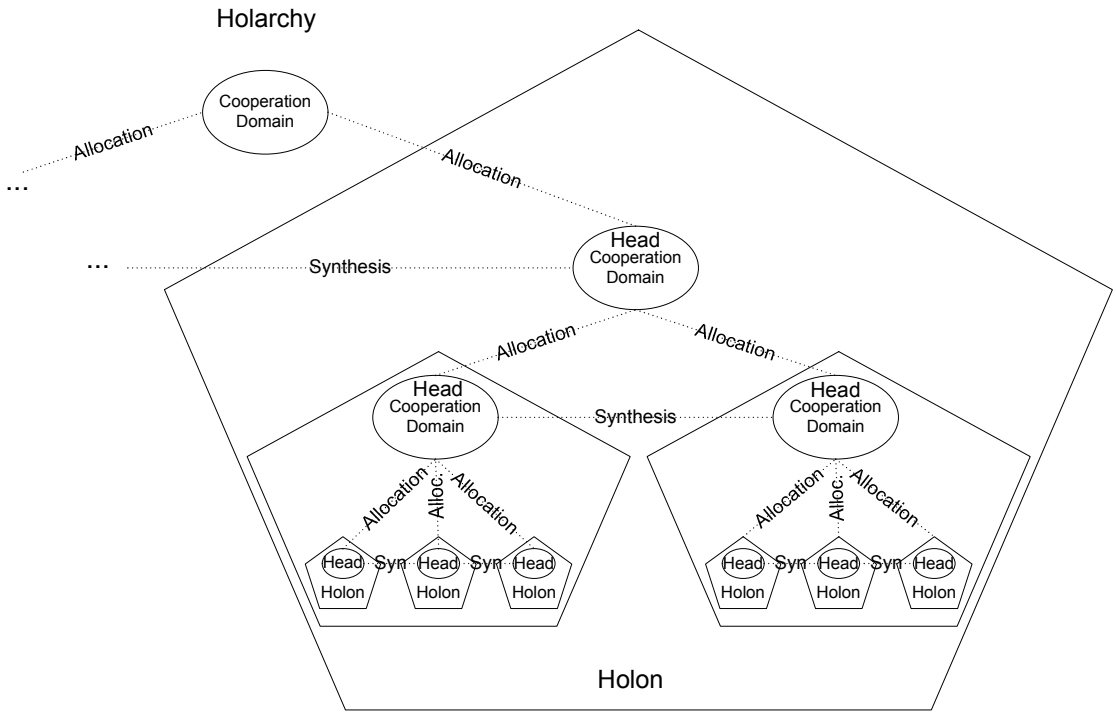

Figura 5.3: Dominios de cooperación

de expresiones de dominio (toda expresión de dominio es a su vez un dominio). Este lenguaje debe permitir la correcta identificación de los agentes afectados por una determinada política. Así, el lenguaje debe proporcionar la suficiente semántica como para poder expresar el hecho de que un sub-dominio no es necesariamente un subconjunto del dominio padre, por lo que los agentes incluidos en dicho sub-dominio no son necesariamente miembros directos del dominio padre, sino miembros indirectos. Esta distinción resulta clave a la hora de determinar el alcance de una determinada política (un sólo subdominio, todos los subdominios, etc.). La definición del lenguaje es:

$$\ell_{dom} \equiv r \in \Re \mid d^{\circ} \mid d^{\bullet} \mid d^{n} \mid \delta_1 + \delta_2 \mid \delta_1 \oplus \delta_2 \mid \delta_1 \times \delta_2 \mid \delta_1 \otimes \delta_2 \mid \delta_1 - \delta_2 \mid (\delta) \mid \delta^{-}$$

Donde r denota un rol, d° representa el conjunto de todos los miembros de un dominio que no son a su vez dominios (entradas de un dominio representando sub-dominios), mientras que $d^{\bullet}$ incluye además los miembros de subdominios anidados recursivamente a d y d^{n} permite acotar dicho anidamiento en una profundidad n.

EdD	Expresión DL	Comentario
$\delta_1 + \delta_2$	$\delta_1 \sqcup \delta_2$	Unión de EdD
$\delta_1 \times \delta_2$	$\delta_1 \sqcap \delta_2$	Intersección de EdD
$\delta_1 \oplus \delta_2$	$(\delta_1 \sqcup \delta_2) \sqcap (\delta_1 \sqcap \delta_2)^- \equiv (\delta_1 \sqcap \delta_2^-) \sqcup (\delta_1^- \sqcap \delta_2)$	Unión disjunta de EdD
$\delta_1 - \delta_2$	$\delta_1 \sqcap \delta_2^-$	Diferencia de EdD
$\{r\}$	{r}	Enumeración
D°	$\{a \mid \mathcal{A} \vDash D(a)\}$	Extensión de una EdD
$D^\bullet$	$\{a \mid \mathcal{A} \vDash D(a) \vee (\mathcal{A} \vDash D'(a) \wedge D' \sqsubseteq D)\}$	Extensión no limitada
D^n	$\{a \mid \mathcal{A} \vDash D(a) \vee (\mathcal{A} \vDash d' \wedge D' \sqsubseteq^j D), 1 \leq j \leq n\}$	Extensión limitada

Cuadro 5.1: Expresiones de dominio en lógica descriptiva

δ^- representa el complementario del dominio δ. $\delta_1 + \delta_2$ es el conjunto que contiene todos los miembros distintos de la unión de los conjuntos resultantes de evaluar las expresiones de dominio δ_1 y δ_2. Del mismo modo, $\delta_1 \times \delta_2$ representa la intersección, $\delta_1 \oplus \delta_2$ representa la unión disjunta ($\delta_1 \oplus \delta_2 \equiv (\delta_1 + \delta_2) - (\delta_1 \times \delta_2)$), y representa $\delta_1 - \delta_2$ la diferencia ($\delta_1 - \delta_2 \equiv \delta_1 - (\delta_1 \times \delta_2) \equiv \delta_1 \times \delta_2^-$).

El cuadro 5.1 recoge la expresión del lenguaje ℓ_{dom} en lógica descriptiva, así como los servicios de inferencia DL que permiten evaluar las expresiones de dominio resultantes.

5.4.2. Descomposición Funcional de un holón

Funcionalmente, un holón de gestión consta de un sistema de control (cabeza) y un sistema de procesamiento (base). El sistema de control realiza las funciones de interfaz colaborativa y control del propio holón. Esta funcionalidad está desempeñada por un agente atómico (o un sistema multiagente) denominado agente de interfaz, representante o mediador, que representa al holón cara al resto de la sociedad de agentes en todos los procesos de interacción. Un agente miembro del holón (en general su cabeza) desempeña el rol de coordinador. Este agente toma la responsabilidad de la meta global y la descompone en tareas parciales adecuadas a la configuración del holón. Su objetivo es encontrar una descomposición que permita una asignación "óptima" de las tareas parciales a los holones asociados (si bien se contempla también una posible (re)configuración del holón "óptima" para las tareas parciales resultantes). La meta global del holón se descompone así en tareas parciales que son asignados a sus unidades organizativas (holones o agentes atómicos).

Los elementos del sistema de control son:

Interfaz inter-holón: Este interfaz gestiona la comunicación con otros holones.

Consta de elementos que permiten al holón negociar y cooperar con otros holones. También incluye facilidades de apoyo a las interfaces de los dominios de cooperación a través de un sistema de comunicación y cooperación.

Interfaz de administración: que enlazan con administradores, supervisores, operadores y personal de mantenimiento. Puede incluir interfaces gráficas de gestión, diagnóstico y explicación.

Control: Responsable de la ejecución de los planes que permiten alcanzar las metas que son responsabilidad del holón. Para ello debe integrar correctamente las capacidades de los miembros del holón y monitorizar su operación.

Intermediario (broker): entre el exterior y los elementos del holón en procesos de negociación, asignación de tareas, distribución de políticas (en jerga de PBN, este agente sería un PDP).

Gateway: Responsable de la interfaz con el elemento gestionado (en jerga de PBN, este agente sería un PEP).

Por su parte, cada componente del sistema de procesamiento del holón está representado por un agente (bien un agente atómico, bien un holón).

5.4.3. Descomposición, asignación y síntesis de tareas

Como se ha comentado anteriormente, las dos principales funciones de la cabeza de un holón son las de actuar como interfaz de cooperación con el exterior y como intermediario y elemento de control hacia el propio holón. En este último contexto, su tarea fundamental consiste en (1) establecer una cadena de competencias necesarias para conseguir la meta deseada, (2) encontrar agentes (posiblemente holones) adecuados a dichas competencias de entre las bases de datos de diversos dominios de cooperación, (3) seleccionar de entre éstos los que mejor se ajusten a los requisitos e (4) integrar correctamente sus capacidades y monitorizar su operación.

Descomposición: Particionamiento adecuado de la meta global en tareas parciales o submetas (supone la identificación de estas últimas).

Asignación: Asignación de las tareas parciales o submetas a un conjunto de agentes (posiblemente holones).

Síntesis: Realización de las tareas parciales o submetas y recopilación.

Para determinar la asignación óptima, el agente coordinador utiliza un protocolo de negociación por medio del cual obtiene las ofertas de todos los holones interesados en llevar a cabo las tareas parciales o submetas. Los holones seleccionados tienen (recursivamente) la posibilidad de actuar a su vez como coordinadores, descomponer de nuevo la submeta asignada y asignar las tareas parciales resultantes a nuevos holones. Se crea así una heterarquía de dominios de cooperación. Nótese que la descomposición y asignación óptima realizada en un determinado nivel puede depender de las decisiones y posibilidades de descomposición y asignación de los niveles más profundos, por lo que puede resultar adecuado utilizar protocolos de compromiso por niveles [SL95, SL01] que ofrecen un elevado grado de paralelismo en la configuración o protocolos más sencillos que retrasen la fase de compromiso y que contemplen retractos y subprotocolos de compensación. En todo caso deben ofrecerse aproximaciones que contemplen la evaluación de diferentes posibilidades en la descomposición y asignación de tareas.

Finalmente, la síntesis de las tareas parciales distribuidas entre los holones miembro deben ser conducidas por negociaciones bilaterales. El modelo de información contendrá una base de conocimiento formal acerca de estrategias de negociación y gestión que serán utilizadas en la descomposición, asignación y síntesis de tareas.

La figura 5.3 mostraba la relación entre estas fases y la composición de una holarquía, y la figura 5.4 muestra la relación existente entre las definiciones de interfaz de agente y las holarquías. En esta última figura se aprecia como cada rol presenta una interfaz cooperativa que identifica los protocolos de interacción admitidos y las políticas asignadas en términos de responsabilidades y permisos. Del mismo modo, las relaciones existentes entre dos o más roles identifican los protocolos de interacción y las políticas implicadas en las mismas. La figura muestra como la interacción entre los holones que conforman las organizaciones ocurre únicamente a través de los roles exclusivos que actúan como cabeza de holón. Puede decirse por tanto que la relación *Relacion 1-2* existente entre los roles *Rol C* y *Rol D* determina la relación existente entre ambos holones (y por tanto ambas organizaciones).

5.5. Caso de aplicación: Gestión holónica de los aspectos de seguridad de un SLA

Se desarrolla a continuación un caso de aplicación del modelo de organización presentado con el fin de demostrar el uso de roles, holónes y dominios de cooperación en la configuración de un modelo de organización adecuado para la gestión de acuerdos sobre nivel de servicios (SLA, del inglés *Service Level Agreement*) en entor-

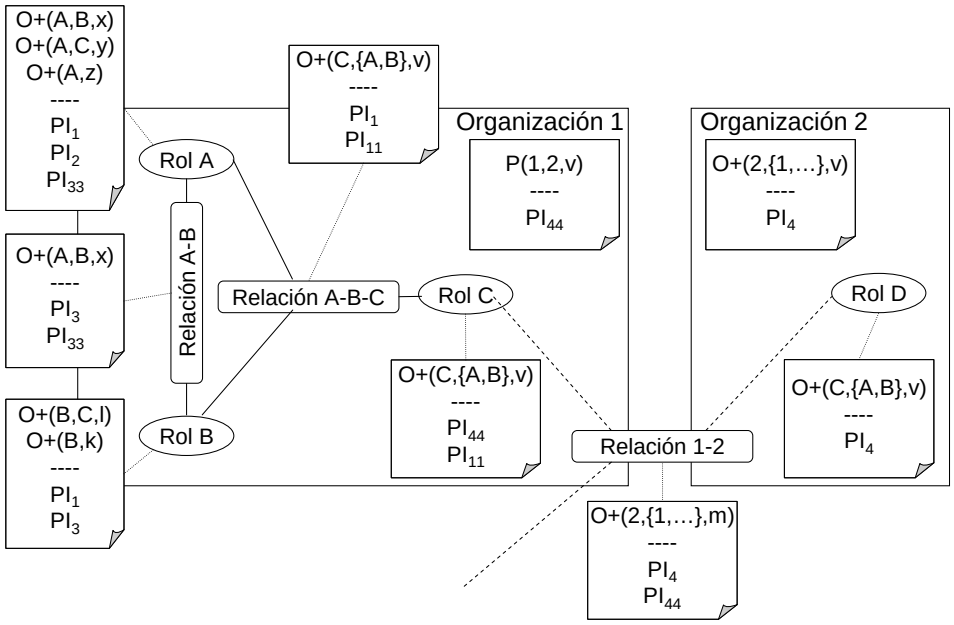

Figura 5.4: Interfaces de cooperación holónicas

nos distribuidos. Para ello se han tomado prestadas las ideas de un caso de estudio presentado en [HAN98] y referido a la garantía de la calidad de servicio desde la perspectiva de la seguridad. El caso de estudio contempla una red de telecomunicaciones formada por un conjunto de nodos de conmutación interconectados a través de una red digital. La red se divide en una serie de redes regionales consistentes en un conjunto de redes de área local interconectadas a su vez mediante una red digital. La figura 5.5 muestra una de estas regiones.

El objetivo de la gestión es monitorizar todas las operaciones ejecutadas en los nodos de conmutación de la red digital con el fin de poder detectar y trazar cualquier tipo de ataque de seguridad. Para ello, los nodos de conmutación generan información de *log* especializada que es almacenada en ficheros de log locales. Estos ficheros no tienen por qué seguir la misma estructura (y por tanto modelo de información) en todas las regiones. Un conjunto de agentes con conocimiento de la estructura de esta información de *log* se encargan de recoger la información de gestión proveniente de los nodos de conmutación a través de encaminadores RDSI y uniformar su estructura antes de almacenarla en un fichero de *log* interoperable. Los agentes recolectores que forman parte de la misma región están interconectados a través de una red de área local Ethernet. La información recogida en los diferentes *log* locales se transfiere sobre un anillo FDDI a un servidor de archivo central. A partir de esta información y de la especificación de patrones de ataque generada por un agente

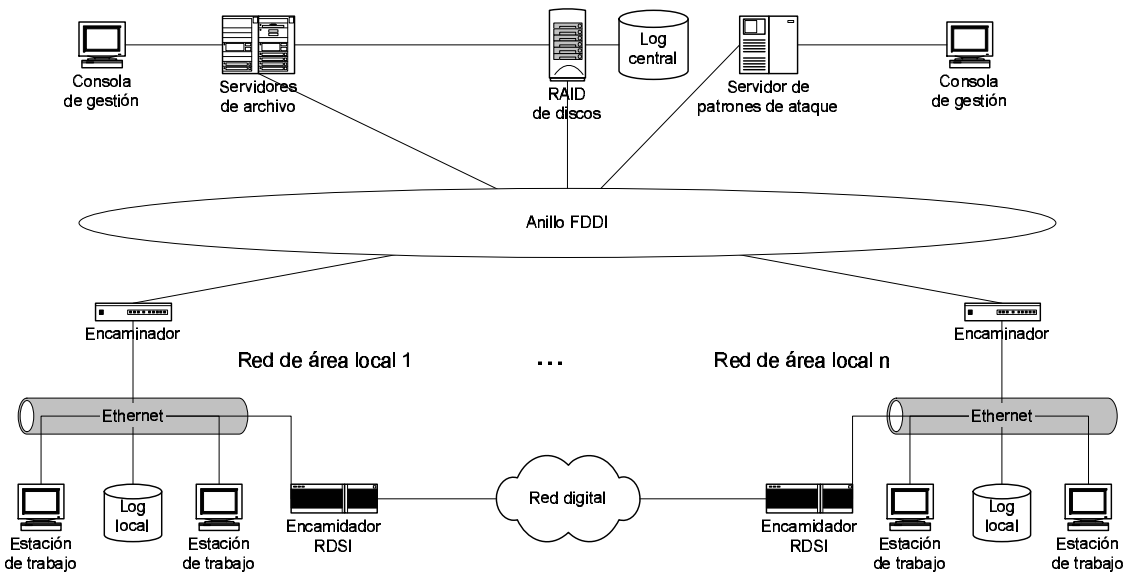

Figura 5.5: Arquitectura de un sistema para control de seguridad basado en SLA

de evaluación asignado a la región, se intentarán identificar los posibles ataques de seguridad ocurridos en la red digital. La función de análisis de la información de log ocurre en cada una de las redes locales, así como en la red regional.

Los privilegios de acceso a los diferentes servidores de archivo implicados en este escenario, así como las responsabilidades asignadas a cada agente implicado en el mismo, se especifican en términos de holones de gestión y, más concretamente, en términos de la configuración de cada uno de estos holones. La configuración de un holón se especifica a través de un conjunto de roles y un conjunto de relaciones existentes entre estos roles. Cada rol tiene asignado un conjunto de políticas y un conjunto de protocolos de interacción que definen respectivamente sus responsabilidades, sus permisos y su interfaz de comunicación. Esto mismo ocurre también con el propio holón, a través de su rol cabeza o interfaz[3]. La figura 5.6 muestra el modelo de organización desarrollado:

- El rol *InspectorSeguridad* está definido tanto para la red regional, como para cada una de las redes locales. Este rol tiene permitido el acceso al log correspondiente a su dominio de cooperación. Además, este rol puede interactuar con el rol *GestorEvaluación* con el propósito de notificarle los ataques detectados, así como con el rol *Recolector* a través de una relación de *Inspección* que

[3]Nótese que, como se ha comentado anteriormente, las relaciones existentes entre los holones de gestión vienen determinadas implícitamente por las relaciones existentes entre los roles que actúan como cabeza de estos últimos

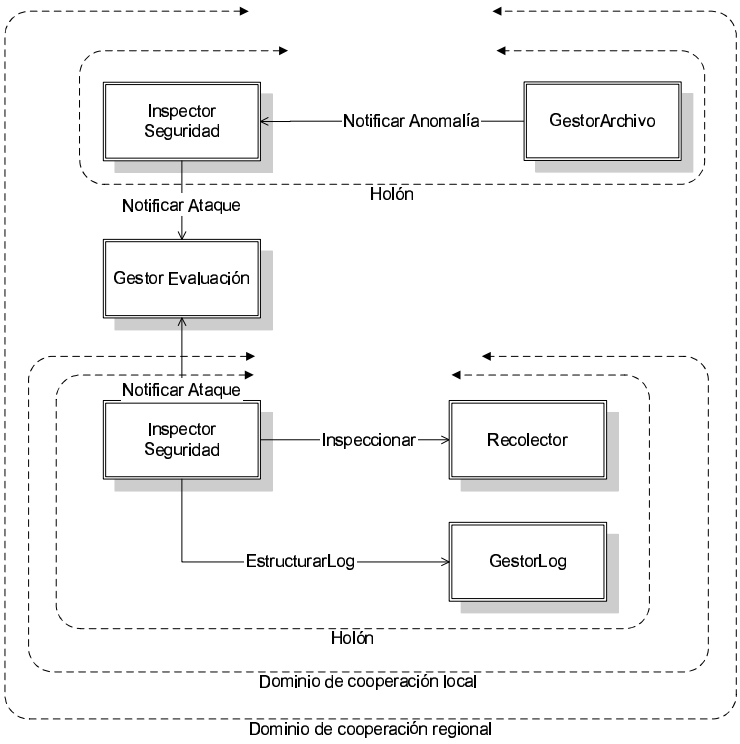

Figura 5.6: Modelo de organización para un sistema para control de seguridad basado en SLA

implica la recolección de datos y con el rol *GestorLog* a través de una relación *EstructurarLog*.

- El rol *GestorEvaluación* es responsable de la especificación de los patrones de ataque, así como de la planificación de las medidas a tomar ante la detección de un ataque. Para desempeñar sus responsabilidades, el *GestorEvaluación* interactuará con el rol *InspectorSeguridad* con el propósito de obtener información acerca de los ataques detectados.

- El rol *GestorArchivo* es responsable de la homogeneización y el almacenamiento estructurado de la información de log en el repositorio central. Este rol puede interactuar ocasionalmente con el rol *InspectorSeguridad* asignado al dominio regional en el caso de que detecte alguna disfunción en el servidor de archivo.

- El rol *Recolector* se ubica en cada uno de los dominios locales y tiene permiso para acceder a la información de log proporcionada por los nodos de conmutación. Este rol interactúa con el rol *InspectorSeguridad* asignado a su mismo dominio local a través de una relación de *Inspección*.

- El rol *GestorLog* está presente en cada dominio local y puede ser desempeñado

por más de un agente en función de las necesidades. Este rol tiene autorizado el acceso a la información de log con el fin de establecer su estructura. Su actuación viene determinada por el rol *InspectorSeguridad* a través de la relación *EstructurarLog*.

El dominio de cooperación local modela la configuración de los holones de gestión (en términos de roles y relaciones implicados) presentes en una red de área local y debe ser por tanto instanciado en cada una de estas redes. Se define además un segundo dominio de cooperación que modela la configuración de una red regional. En términos de modelo, este último dominio incluye como subdominio al primero. En términos de instancia, una misma instancia del dominio regional contendrá como subdominios un conjunto de instancias del dominio local.

La figura 5.7 muestra como la utilización del concepto de holón permite replicar esta estructura en el caso de que, con el propósito de mejorar la escalabilidad de la solución, se decida subdividir la red regional y utilizar para cada una de las nuevas subredes creadas la misma estructura que para la red regional inicial, o bien se decida integrar la gestión de otras redes regionales. El nuevo dominio regional permite integrar esta estructura como un holón local ya que en ambos casos se presenta la misma interfaz de cooperación (definida por el rol *InspectorSeguridad* y la relación *NotificarAtaque*). Nótese que en la figura sólo aparecen reflejados los roles que participan en la interacción inter-holónica.

En el apéndice A se recoge la especificación en lógica descriptiva del modelo de organización planteado en este caso de aplicación. A partir de esta especificación, los agentes de gestión pueden razonar de forma automática acerca de la estructura organizativa del entorno en que participan, realizando inferencias sobre los roles existentes, sus propias relaciones con éstos roles, etc. De este modo se facilita la automatización de las tareas de gestión. En dicho apéndice se recogen dos posibles soluciones, fruto de la posibilidad de interpretar el modelo de dos formas diferentes

5.6. Políticas sociales

En el proceso de conceptualización de una sociedad de agentes se hace preciso capturar y modelar la *conciencia social* de los agentes que participan en la misma. La propuesta de modelado de esta conciencia social elaborada en esta tesis, se centra en el concepto de *política social*. En el contexto de este trabajo de tesis, una *política* se define como una especificación formal y declarativa, derivada de las metas globales de una sociedad de agentes, de una regla que define alternativas en el comportamiento de los agentes que participan en la misma y que las interpretan. Para formalizar

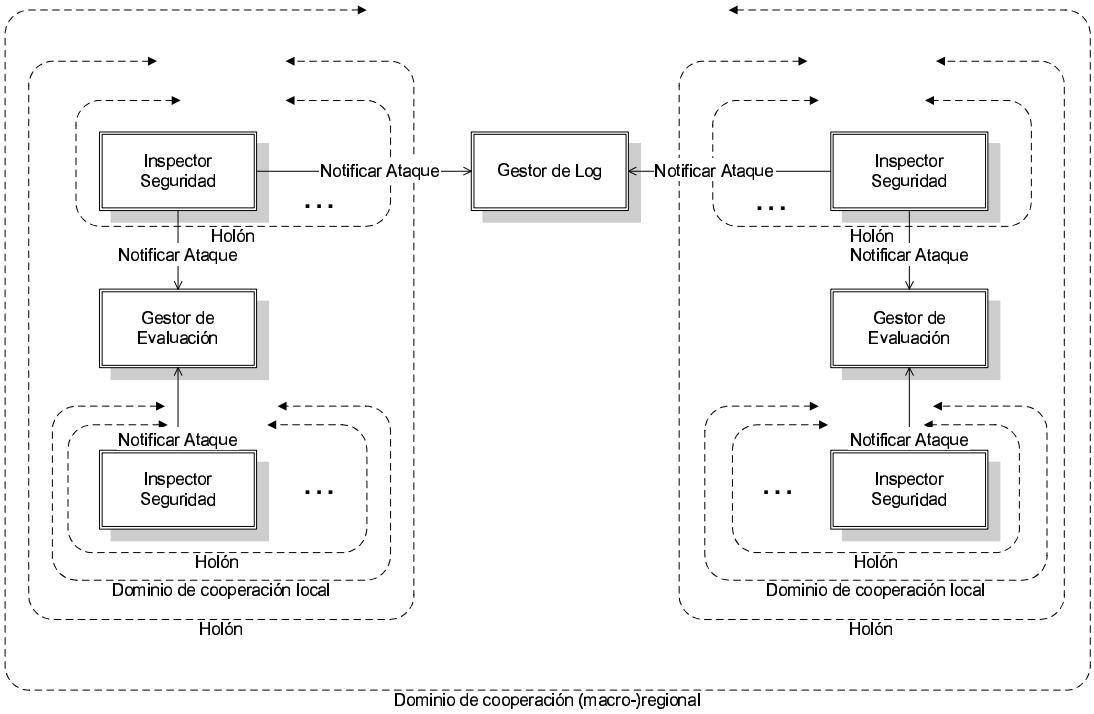

Figura 5.7: Modelo de organización para un sistema para control de seguridad basado en SLA (II)

este concepto se han utilizado cuatro modalidades deontológicas interrelacionadas: normas, obligaciones, responsabilidades y autorizaciones).

La autonomía intrínseca a los agentes, junto con el carácter abierto de las sociedades de agentes, hace que parezcan poco plausibles propuestas basadas en restringir el comportamiento de los agentes en tiempo de diseño. La representación explícita de forma declarativa de las normas, responsabilidades y autorizaciones, permitirá su distribución y posterior interpretación por parte de los agentes implicados por las mis-mas. Se flexibiliza así el diseño tanto de los agentes como de la propia infraestructura social y se facilita la introducción de cambios en el comportamiento y la estrategia de los agentes (y de la sociedad en último termino) mediante la simple modificación de dichas políticas, sin necesidad de recodificar dichos agentes ni la infraestructura social (y por tanto los servicios de agencia). La sociedad puede así adaptarse dinámicamente a cambios en los requerimientos deshabilitando políticas o reemplazándolas por otras nuevas sin necesidad de interrumpir su funcionamiento.

Se describen a continuación las cuatro modalidades indicadas:

5.6.1. Normas

En general, una norma representa un comportamiento específico que es visto como beneficioso bien por una organización concreta, bien por la sociedad en su conjunto. La especificación formal de un conjunto de normas permitirá influenciar el comportamiento de los agentes que se adhieran a las mismas. Esto ayudará a estandarizar y a coordinar el comportamiento de los diferentes agentes que participan en una sociedad abierta, permitiéndoles anticipar el comportamiento del resto y facilitando con ello sus procesos de cooperación, negociación e interacción en entornos heterogéneos [Dig00b].

Las normas no restringen la autonomía de los agentes, si bien pueden influir en sus intenciones, facilitando su coordinación y haciéndola más eficaz y eficiente. Un agente puede decidir adoptar una norma o no en base tan solo a los beneficios que observe en la adopción de dicha norma, siguiendo así un comportamiento que denominaremos *socialmente responsable*.

Se introduce un nuevo operador modal N para especificar las normas:

$N^D(\alpha \mid \psi)$ para indicar que en el dominio D es una norma realizar la acción α cuando ψ es verdad.

$N^D(\phi \mid \psi)$ para indicar que en el dominio D es una norma que la fórmula ϕ sea verdad cuando ψ es verdad.

Un típico ejemplo de norma presente en una sociedad de agentes sería:

$$\forall a_1, a_2 \in S[request(a_1, a_2, \_)]N^S(response(a_2, a_{1,}) \mid true)$$

que hace uso de la lógica dinámica [Har87] y el nuevo operador N para expresar que todo agente debería contestar las solicitudes efectuadas por otro agente siempre que sea posible.

5.6.2. Obligaciones

Las obligaciones representan políticas de asignación de responsabilidades en la (no) realización de acciones. Constituyen por tanto un mecanismo explícito que permite influenciar el comportamiento de los agentes sin anular con ello su autonomía, aunque sí reduciéndola. La representación explícita de las obligaciones y la existencia de un framework adecuado para su realización permitirá a los agentes interpretar sus responsabilidades para con la organización en que actúan y llevar a cabo un razonamiento deliberativo acerca de las mismas.

Al igual que ocurre con las normas, las obligaciones permiten también influenciar el comportamiento de los agentes. Sin embargo, conviene diferenciar ambos conceptos: mientras que las normas no restringen la autonomía de los agentes, ya que tan sólo les informan de comportamientos estándares que resultan socialmente beneficiosos, las obligaciones sí restringen la autonomía de los agentes, en la medida en que representan sus responsabilidades dentro de la organización en que desarrollan sus actividades. Esta merma en la autonomía es en pro de un comportamiento más coherente y equilibrado con respecto al esperado por la organización, y que denominaremos *comportamiento orgánico*.

Para expresar las obligaciones, se introduce un nuevo operador modal O. El alcance de una obligación, al contrario de cómo ocurría con las normas, suele estar circunscrito a un único rol que figura como target de la misma, y puede ser originada por un rol concreto.

$O^D_{t,s}(\alpha \mid \psi)$ para indicar que el rol t está obligado a realizar la acción α cuando ψ es verdad. Cada rol se asocia al dominio D en que participa, de modo que sea más sencillo organizar y analizar tanto los propios roles, como las obligaciones asociadas a los mismos. s denota el rol que ha originado la obligación (subject). Si la obligación viene impuesta por la organización a la que representa el dominio D, o por la propia sociedad ($D = S$), solo aparecerá $O^D_t(\alpha \mid \psi)$. D es una expresión de dominio que representa la organización/sociedad responsable de forzar la penalización en caso de violación.

$O^D_{t,s}(\phi \mid \psi)$ para indicar que el rol t está obligado a considerar ϕ verdad cuando ψ es verdad.

Generalmente, el comportamiento orgánico esta también muy influenciado por las normas sociales, por lo que tiene un componente de comportamiento socialmente responsable. De hecho, nuestro modelo define agente orgánico como aquel agente cuyo comportamiento está influenciado por un conjunto de obligaciones que representan sus responsabilidades dentro de la organización en que participa y cuyo diseño le predispone a seguir las normas establecidas por la sociedad en que actúa. En este sentido, una norma social típica en toda sociedad de agentes es aquella que dictamina que se deben cumplir las obligaciones siempre que sea posible, tal y como expresa:

$$\forall a \in S, d \in D(S) \cdot N^s(\alpha \mid O^d_a(\alpha \mid \psi) \wedge \psi)$$

Siguiendo la lógica deontológica tradicional [Meyer93][Wright59], se contemplan tres modalidades de obligaciones: la Obligación (Obligation) propiamente dicha,

que representa la responsabilidad de un agente en la realización de una determinada acción, la Prohibición (Interdiction) $I^D_{t,s}(\alpha \mid \psi)$, que permite expresar la responsabilidad de un agente en la no realización de una determinada acción, y el Permiso (Permission) $P^D_{t,s}(\alpha \mid \psi)$, que expresan explícitamente la ausencia de una determinada prohibición.

Las dos últimas modalidades pueden expresarse en función de la primera haciendo uso del concepto de negación de acción proveniente de la lógica de acción. Así, se tiene que $I(\alpha) \leftrightarrow O(\overline{\alpha})$ y $P(\alpha) \leftrightarrow \neg I(\alpha) \leftrightarrow \neg O(\overline{\alpha})$ pero se introduce un nuevo operador modal para cada una de ellas en aras de una mayor expresividad. Algunos autores introducen incluso un nuevo operador para representar el concepto de discrecionalidad o libertad $D(\alpha) \leftrightarrow \neg O(\alpha)$.

5.6.3. Violación de obligaciones

Si se desea conservar la autonomía de un agente orgánico, deben contemplarse situaciones en las que las restricciones normativas de tipo obligativo puedan ser violadas. La aproximación deontológica permite cierto grado de libertad de elección, a la vez que permite asociar explícitamente consecuencias a la toma de decisiones. La decisión de obedecer una norma está supeditada a un proceso deliberativo de toma de decisiones guiado únicamente por la ponderación de la relación coste/beneficio (individual y social) asociada al seguimiento de la misma. Sin embargo, el cumplimiento de una obligación está sujeto a un proceso deliberativo de toma de decisiones guiado además por una ponderación de las consecuencias derivadas de la violación de dicha obligación, esto es, del coste asociado a la sanción que acarrea el incumplimiento de la obligación. Por su parte, la interpretación de las autorizaciones (positivas y negativas) no está sujeta a ningún proceso deliberativo de toma de decisiones, ya que en ningún caso pueden ser violadas.

La relación de ponderación del beneficio social y la relación de ponderación de consecuencias derivadas de la violación de obligaciones definirán un orden parcial entre las normas y las obligaciones respectivamente. El razonamiento basado en obligaciones resulta más sencillo que el razonamiento basado en normas. Esto es debido a que las obligaciones tienen penalizaciones explícitas asociadas a su violación. Mientras que las consecuencias derivadas de no seguir una norma son indirectas y a más largo plazo. La reducción de la lógica deontológica a la lógica dinámica permite introducir el concepto de violación como un nuevo predicado en la formalización de los operadores deontológico, así:

- $O_{i,j}(\alpha(i)) = [\overline{\alpha(i)}]V^k_{i,j}(\alpha)$ expresa que una acción es obligada si su no realización conlleva una violación,

- $F_{i,j}(\alpha(i)) = [\alpha(i)]V^k_{i,j}(\alpha) \equiv O_{i,j}(\overline{\alpha(i)})$ expresa que una acción es prohibida si su realización conlleva una violación y, por último
- $P_{i,j}(\alpha(i)) = \neg[\alpha(i)]V_{i,j}(\alpha) \equiv \neg O_{i,j}(\overline{\alpha(i)}) \equiv \neg F_{i,j}(\overline{\alpha(i)})$ expresa que una acción es permitida si su realización no conlleva una violación. También se suele utilizar una representación más restrictiva que, junto con un predicado only(), permite evitar la paradoja de la libre elección. Esta última connotación se conoce como permiso contextual.

Como puede observarse en las definiciones anteriores, para cada acción se define uno o más estados de violación $V^k(\alpha)$, uno por cada una de las razones por las que la (no) ejecución de la acción es incorrecta. El motivo por el que se definen estos estados de violación y se representa la razón de la violación en el predicado, es permitir expresar la acción correctiva.

Las siguientes ecuaciones muestran la utilización del predicado de violación:

$$[AddSLA(r : RA, C : C, gold : P)]O^{Intranet}_{RA,C}(Fwd(r : RA, c : C, t : T))_{before(10ms)}$$

$$V^{Intranet}_{RA,C} : Fwd(r : RA, c : C, t : T)_{before(10ms)} \rightarrow I^{Intranet}_{L,R}(Charge(r : RA, c : C, 25\,\%))$$

$$V^{Intranet}_{RA,C} : Fwd(r : RA, c : C, t : T)_{before(10ms)} \rightarrow [Fwd(r : RA, c : C, t : T)_{bfr(10ms)}]P^{Intranet}_{RA,C}(Charge(r : RA, C : C, 100\,\%))$$

RA representa el rol *Router de Acceso* en una red MPLS, C el rol *Cliente* y P el dominio que clasifica los clientes en $\{gold, silver, bronze\}$. La primera ecuación expresa que, tras especificarse en un acuerdo sobre nivel de servicio la pertenencia del cliente c al tipo *gold*, de cara a la operación del router de acceso r, dicho router deberá retransmitir todo el tráfico con origen c en un tiempo inferior a 10ms. El ámbito de aplicación de esta obligación es la Intranet de la empresa. Si en algún momento el router de acceso no pudiese retransmitir el tráfico del cliente c en 10ms, la segunda ecuación determina que se habrá incumplido el acuerdo (a partir de la definición axiomática de obligación) y no se podrá cobrar el 25 % del contrato en tanto no se reestablezca el SLA, tal y como especifica la tercera ecuación.

Este ejemplo muestra también cómo, pese a que las obligaciones estén referidas a roles o dominios, a menudo es necesario ligarlas a un agente o elemento concreto perteneciente a dicho rol o dominio. Dicha ligadura viene representada por el operador denotado por ':'.

Pese a que, como hemos visto, la especificación de ciertas obligaciones y violaciones conlleva la necesidad de ligar algunos de los roles implicados a instancias concretas de dichos roles, sigue resultando interesante referir dichas obligaciones a roles con el fin de facilitar su distribución.

5.6.4. Obligación y Responsibilidad

Un concepto que juega un papel importante en el modelado de la conciencia social de los agentes es el de responsabilidad. Dentro de nuestro marco de trabajo de agentes, identificamos el concepto de responsabilidad con el de obligación, y por tanto si un agente tiene la obligación de realizar una acción, se dice que es responsable de dicha acción. Esta obligación no tiene por que ser hacia otro agente (e.g. obligaciones creadas mediante una relación de poder existente entre ambos), también puede ser una obligación propia, y por tanto se dice que es una responsabilidad intrínseca del rol. Esto puede utilizarse para modelar mediante obligaciones las responsabilidades asignadas a un agente como resultado de su posición en la organización. Las obligaciones representan por tanto políticas de asignación de responsabilidad en la realización de acciones. El concepto de responsabilidad se modela explicitando a quién afecta la sanción que surge como consecuencia de la violación de la obligación a la que se refiere dicha responsabilidad. Cuando un agente no realiza una acción respecto de la cual tenía contraída una obligación (y por tanto es responsable), es misión suya reparar la situación, por lo que dicho agente pasa a ser el sujeto de la violación y por tanto de la acción o situación implicada por la misma.

El concepto de responsabilidad resulta de especial interés cuando se quiere modelar una situación en que un agente tiene contraída una obligación pero realmente tiene otro agente subordinado realizándola. Sin el concepto de responsabilidad, la obligación contraída por el agente subordinado sólo se referiría a dicho agente, por lo que sería el único sujeto de la sanción derivada de la no realización de la acción obligada. Como se verá a continuación al estudiar la dinámica de las obligaciones, el concepto de responsabilidad permite separar el sujeto de la sanción del sujeto de la obligación, y asociarlo al sujeto de la responsabilidad (que no siempre es el sujeto de la obligación). Se dice entonces que el primer agente ha delegado la responsabilidad de la realización de dicha acción, pero no ha delegado el control.

5.6.5. Separación de responsabilidades

La técnica de separación de responsabilidades constituye un ejemplo paradigmático de política obligativa que implica la utilización conjunta de permisos y prohibiciones en la especificación de comportamientos normativos ante actividades po-

tencialmente conflictivas. En función de cómo se realice dicha separación, puede distinguirse entre separación dinámica y separación estática de responsabilidades.

En la separación dinámica de responsabilidades todos los agentes que actúan en un mismo puesto tienen permitida inicialmente la realización de un conjunto de acciones potencialmente conflictivas, pero tras la realización de una de estas acciones, se les prohíbe la realización del resto de acciones conflictivas. Las siguientes restricciones muestran cómo se permite a los agentes que actúan como *Gestores* (G) en el dominio *Contabilidad*, emitir (primera restricción) y autorizar (segunda restricción) *Cheques* (C), pero se prohíbe que un mismo agente firme y autorice el mismo cheque.

$$P_G^{Contabilidad}(emitir(m:M, c:C) \mid m.id \neq c.autorizadorID)$$
$$P_G^{Contabilidad}(autorizar(m:M, c:C) \mid m.id \neq c.emisorID)$$

Se observa de nuevo cómo las obligaciones se expresan en términos de dominios y roles, pero las condiciones necesitan de su ligadura a elementos concretos. Si las condiciones se expresasen como igualdades, los permisos pasarían a ser prohibiciones.

La separación (estática) de responsabilidades representa un concepto más restrictivo, en el que se prohíbe a ciertas posiciones la realización de tareas conflictivas. El siguiente ejemplo muestra cómo se separa en dos roles (gestor -G- y representante -R-) las responsabilidades de emitir (primera restricción) y autorizar (segunda restricción) cheques en el dominio Contabilidad.

$$F_{Cont}^{G}(emitir(M,C) \mid true) \wedge P_{Cont}^{R}(emitir(R,C) \mid true)$$
$$P_{Cont}^{G}(autorizar(G,C) \mid true) \wedge F_{Cont}^{R}(autorizar(R,C) \mid true)$$

Se observa como la separación estática de responsabilidades no requieren ligaduras a elementos concretos. Esto suele implicar restricciones sobre grupos de políticas para restringir la especificación de políticas que pueden entrar en conflicto. Para establecer estas restricciones se utiliza el concepto de meta-política.

A partir de los teoremas provenientes de la lógica expuesta, se pueden elaborar métodos de propagación de restricciones, como el descrito en [BGM98], que permitirán a los agentes inferir nuevas obligaciones, permisos y prohibiciones a partir de las dadas.

5.6.6. Autorizaciones

Las políticas de autorización definen el conjunto de peticiones aceptables por un elemento del sistema en base a la identidad del solicitante y a las circunstancias (condiciones) en que se produce la solicitud.

Las autorizaciones se conceden en base a la identidad del solicitante, por lo que tienen estrecha relación con las políticas de autenticación del sistema, si bien pueden (y deben) expresarse con independencia de estas últimas.

Con el fin de hacer más escalable el sistema de políticas, y al igual que ocurría con las dos modalidades anteriores, los sujetos y destinatarios de las autorizaciones se expresan en términos de roles y dominios, para lo cual se hace uso del lenguaje ℓ_{dom} introducido en la sección 5.4.1.1.

A pesar de lo que pudiese parecer en un primer momento, los conceptos de autorización positiva y negativa no son equivalentes a los de permiso y prohibición, respectivamente. La principal diferencia recae en el hecho de que las obligaciones (y por tanto los permisos y prohibiciones) son interpretadas por los sujetos de las mismas, y por tanto es posible su violación, mientras que las autorizaciones se interpretan por mecanismos de control de acceso ligados al destinatario, no siendo posible su violación. Por otra parte, las autorizaciones no restringen la autonomía de los agentes, si bien influyen en sus procesos de toma de decisiones al ser prerrequisitos para la realización de determinadas acciones.

Una autorización negativa se puede interpretar como una prohibición con coste infinito asociado a su violación o bien una prohibición con acción *fallo* asociada a su penalización.

Para representar el concepto de autorización se introducen dos nuevos operadores modales A+ y A-:

$A +_s^D (\alpha \mid \psi)$ expresa que el rol s está autorizado a solicitar la acción α a un objeto perteneciente al dominio expresado por D cuando ψ es verdad. Asociado a D habrá un controlador de dominio encargado de validar las peticiones y que será por tanto el responsable de forzar el cumplimiento de las autorizaciones.

$A -_s^D (\alpha \mid \psi)$ expresa que el rol s no está autorizado a solicitar la realización de la acción α a un objeto perteneciente al dominio expresado por D mientras ψ es verdad.

5.7. Dinámica de las obligaciones

Hay obligaciones que no desaparecen tras la realización de la acción obligada, sino que persisten a lo largo del tiempo. Éste es el caso de las obligaciones asignadas a sus roles por una organización y que modelan comportamientos responsables intrínsecos. Así, una organización que actúa como agencia de viajes (AV) puede requerir que todos los agentes que actúan como comerciales suyos respondan a todas las consultas efectuadas por agentes cliente de confianza:

$$[query(Cliente, Vendedor, \_)]O^{AV}_{Vendedor}(reply(Vendedor, Cliente, \_) \mid trusted(s))$$

Este es un ejemplo de obligación que no desaparece tras responder a la primera consulta, sino que persiste como responsabilidad intrínseca al rol Vendedor.

Sin embargo numerosas obligaciones desaparecen tras la realización de la acción obligada, como es el caso de las obligaciones contextuales creadas por un agente sobre otro a través de una relación de *poder* o de *autorización temporal*. Es necesario por tanto modelar estas dos relaciones para poder formalizar la dinámica de las obligaciones. A continuación describiremos estas dos relaciones y las utilizaremos para formalizar la dinámica (creación, derogación y distribución) de las obligaciones mediante la lógica ilocucional.

5.7.1. Relación de poder

La relación de poder es el mecanismo formal que permite establecer nuevas obligaciones (y creencias), así como eliminarlas. Existe una relación de poder entre un rol i y un rol j con respecto a una acción α, si un agente actuando como i está facultado para ordenar la realización de α a otro agente que esté actuando como j. Para modelarla se introduce el predicado $\Pi^{\alpha}_{i,j}$.

La relación de poder entre dos roles suele tener carácter persistente, ya que representa la estructura normativa de la organización y ésta no tiende a cambiar fácilmente con el paso del tiempo.

Veamos a continuación un uso de la relación de poder para expresar una obligación contraída como consecuencia de una relación de poder entre dos roles:

$$\Pi^{request(pay(t,s,q)}_{s,t} \rightarrow [request(s, t, pay(t, s, q)]O^{D}_{t}(pay(t, s, q) \mid trusted(s))$$

La ecuación expresa que el rol s tendrá que pagar la cantidad q requerida por t cada vez que t se lo solicite, ya que existe una relación de poder entre t y s para efectuar solicitudes de pago.

Sin embargo, si lo que se pretendía expresar con dicha ecuación es el hecho de que t pudiese requerir el pago de la cantidad q una sola vez por ciertos servicios prestados, no sería correcta la representación anterior, ya que entonces sería necesario anular la relación de poder, que es una relación de carácter persistente como se ha comentado anteriormente.

$$O^{D}_{t}(pay(t, s, q) \mid trusted(s)) \rightarrow ([pay(t, s, q)] \rightarrow \neg\Pi^{request(pay(t,s,q))}_{s,t})$$

Resulta más adecuado introducir una relación de poder de carácter temporal que denominaremos relación de autorización temporal.

5.7.2. Relación de autorización temporal

La relación de autorización temporal se establece por tiempo definido, con acuerdo mutuo y bajo ciertas restricciones. Se modela mediante el predicado $\triangle(i, \alpha)$ para indicar que i está autorizado a realizar α. La semántica de esta relación difiere de la semántica de la relación de poder en su carácter transitorio. La relación de potestad se establece por un cierto periodo de tiempo bajo acuerdo mutuo (i.e bajo ciertas restricciones). La principal cualidad de esta relación es que permite establecer la dinámica de obligaciones en base a que puede retraerse. El ejemplo anterior quedaría modelado utilizando la relación de autorización como:

$$\triangle_{s,t}^{request(pay(t,s,q))} \rightarrow [request(s,t,pay(t,s,q)]O_t^D(pay(t,s,q) \mid trusted(s))$$
$$O_t^D(pay(t,s,q) \mid trusted(s)) \rightarrow ([pay(t,s,q)] \rightarrow \neg\triangle_{s,t}^{request(pay(t,s,q))})$$

5.8. Creación, derogación y distribución de políticas mediante actos ilocucionales

Siguiendo las ideas originales de la teoría de actos hablados (*speech acts*) [Aus62, SV85], la comunicación entre agentes no debe ser vista simplemente como una transmisión de información, sino también como acciones que cambian el estado mental de los agentes implicados en la misma (y por tanto sus creencias, deseos e intenciones).

Un aspecto importante de la teoría de actos hablados es qué puede inferirse como resultado de un *acto hablado*. En este sentido, nos centraremos en los *actos ilocucionales*, que son aquellos actos realizados al decir algo, por ejemplo, preguntar o contestar una pregunta, proporcionar información, etc. [SV85] descompone cada acto ilocucional en la fuerza ilocucional, que especifica el tipo de acción (requesting, warning, informing, etc.) realizada y el contenido proposicional, que especifica los detalles de la acción (qué se esta preguntado, acerca de qué se está informando, etc.). Searle descompone a su vez el contenido proposicional en un acto predicativo y un acto referencial, si bien esta última distinción no resulta importante en el contexto de este trabajo de tesis.

Las relaciones de poder y de autorización temporal, junto con la lógica ilocucional, representan la base para introducir nuevas obligaciones, creencias y permisos en el sistema mediante actos de comunicación entre agentes. La existencia de una relación de poder o de autorización es la condición bajo la cual una ilocución prescriptiva se considera que tiene fuerza directiva. En este sentido, pueden crearse, distribuirse y delegarse obligaciones y, también, solicitarse, concederse y delegarse autorizacio-

nes, creándose así un entorno dinámico para el establecimiento y la derogación de normas autorizadas.

Consideraremos tan solo la fuerza ilocucional *directive* ya que será la utilizada para formalizar el concepto de delegación de responsabilidad:

- Las directives (DIR) sirven para requerir la realización de una acción. Se utilizarán, en el contexto de una relación de poder o de autorización temporal, para crear y dar a conocer una obligación en el destinatario del acto ilocucional.

  $([DIR_{\Pi}(i,j,\alpha)]O_{j,i}(\alpha)) \leftarrow \Pi^{\alpha}_{i,j}$

  $([DIR_{\triangle}(i,j,\alpha)]O_{j,i}(\alpha)) \leftarrow \triangle^{DIR(i,j,\alpha)}_{i,j}$

El primero de los axiomas expresa que la existencia de una relación de poder entre los roles i y j es suficiente para que una ilocución directiva emitida por i genere una obligación en j. El segundo axioma expresa que la existencia de una relación de autorización entre los roles i y j para requerir la realización de una acción α es suficiente para que dicho requerimiento cree en j la obligación de realizar dicha acción.

5.8.1. Delegación de responsabilidad

La delegación de responsabilidad modela el acto de confiar la realización de una tarea obligada a otro agente. A continuación se muestra como la dinámica de delegación de una tarea obligada a otro agente se puede modelar mediante el intercambio de uno o más actos de comunicación imperativos y la gestión correcta del predicado de violación.

La delegación de una obligación por parte de un agente implica que éste confía en que el agente delegado actuará en su beneficio, por lo que para que ésta se produzca es necesario que exista algún tipo de vínculo entre el agente que delega la tarea y el agente al que se delega ésta. Este vínculo se concreta en las relaciones de poder, confianza y autoridad: la relación de poder es necesaria para crear una obligación en el agente al que se delega la tarea; la relación de confianza se utiliza para escoger dicho agente entre los posibles candidatos (aquellos respecto de los cuales se tiene una relación de poder); la relación de autoridad se utiliza para transferir las autorizaciones necesarias para poder llevar a cabo la tarea delegada. De este planteamiento se desprende que debe existir un enlace explícito entre los actos de comunicación imperativos y las posiciones normativas establecidas en el modelo de roles de la sociedad. Estas últimas vienen determinadas por las relaciones de poder, autoridad y confianza.

Una vez delegada la responsabilidad en la realización de una tarea, dicha responsabilidad pasa a estar compartida, por lo que quien delega una responsabilidad debe asegurase de que el agente delegado actúa de forma apropiada. El concepto de responsabilidad se modela explicitando a quién afecta la sanción que surge como consecuencia de la violación de la obligación a la que se refiere dicha responsabilidad. Esto debe capturarse en la delegación de responsabilidad. Como hemos visto, a través de las relaciones de poder y confianza, y utilizando un acto de comunicación imperativo (directiva) un agente puede influenciar el comportamiento de otros agentes imponiéndoles obligaciones. La lógica permite formalizar el concepto de delegación de responsabilidad como:

$$O(i,j,\alpha) \rightarrow (\Pi^{\alpha}_{i,k} \wedge T^{\alpha}_{i,k} \rightarrow [DIR_{\Pi}(i,k,\alpha)](O(i,j,\alpha)))$$

Esta formalización expresa el hecho de que la delegación a un tercero k de la responsabilidad contraída por un agente i (posiblemente con otro agente j con el que mantiene una relación de poder) para realizar una tarea, no exime a i de la responsabilidad en su realización. Si la tarea no se lleva a cabo, tanto i como k incurrirán en una violación y se verán afectados por la sanción derivada de dicha violación.

Sin incluir el concepto de responsabilidad, sólo podríamos expresar obligaciones independientes del tipo $O(k,i,\alpha) \rightarrow [\overline{\alpha}]V^{\alpha}_{k,i}$. Por lo que si i es un agente en el que j ha delegado una obligación contraída $O(k,i,\alpha) \rightarrow [\overline{\alpha}]V^{\alpha}_{i,j}$, el agente que realiza la tarea podría ser el único responsable si la tarea no se completase con éxito. La obligación del agente i desaparece, pero es necesario mantener su responsabilidad.

El concepto de responsabilidad y su gestión mediante la delegación de responsabilidad permite expresar $O(i,j,\alpha) \rightarrow [\overline{\alpha}]V^{\alpha}_{j,k}$, lo cual resulta más próximo a la realidad, ya que el agente obligado inicialmente sigue siendo el responsable de que la tarea se complete con éxito (es el sujeto de la violación, y por tanto de la sanción asociada), y el agente obligado puede no ser responsable de la realización de dicha acción. De otra forma siempre sería responsable respecto al agente que le delegó la tarea.

Capítulo 6

Experimentación y Resultados

Índice General

6.1. Diseño Experimental 171
6.2. Marco de trabajo 178
6.3. Resultados 179

Con el fin de demostrar la factibilidad de la implementación y verificar la viabilidad de los diferentes modelos propuestos en este trabajo de Tesis, se ha elaborado un diseño experimental y se han desarrollado una serie de herramientas que, en conjunto, conforman un marco de trabajo para el modelo de arquitectura propuesto sobre el que llevar a cabo los experimentos descritos en dicho diseño. En este capítulo se describe el diseño experimental elaborado, se presenta el marco de trabajo desarrollado para facilitar la experimentación y se recogen los principales resultados obtenidos a partir de dicha experimentación. Los cuatro apéndices incluidos describen pormenorizadamente los resultados obtenidos a partir de este proceso de experimentación.

6.1. Diseño Experimental

El cuadro 6.1 desglosa el conjunto de experimentos a realizar sobre los "mappings" $CIM - OWL_{FULL}$ y CIM-$\mathcal{ALE}\mathcal{CNOQ}^{-}_{\mathcal{HR}^{+\circ}}$ propuestos en el capítulo 3. Dicho cuadro recoge los principales componentes de cada uno de los modelos CIM desarrollados por el DMTF en su versión 2.7, y muestra la complejidad asociada a la formalización de estos modelos en términos de número de clases, número de asociaciones (dependencias, asociaciones y agregaciones), y número de atributos. Siguiendo la filosofía del meta-esquema CIM, las asociaciones se contemplan como un tipo de clase.

Especificación CIM	**Clases**	**Atrib.**
CIM Core Specification. DMTF SysDev-WG		
Elementos lógicos y sus subclases	12	37
Elementos físicos y dispositivos lógicos	4	39
Colecciones y conjuntos de redundancia	3	2
Grupos de redundancia	4	9
Producto, FRU e identidad software	4	28
Estadísticas	7	13
Información de ajuste, perfiles, capacidades y gestión de energía	7	7
Ajustes, configuraciones y parámetros	5	7
Jerarquías de asociación, dependencia y agregación	78	21
CIM User Specification. DMTF User-WG		
Credenciales y autenticación	5	16
Servicios de seguridad	7	1
Servicios de seguridad Kerberos	2	4
Servicios de seguridad de clave pública	4	12
control de acceso	3	14
Organización y persona	8	65
Grupos y roles	4	41
Cuentas	2	12
Jerarquías de dependencia y agregación	45	11
CIM Database Specification. DMTF Database-WG		
Entorno de base de datos	6	13
Software y estadísticas	3	21
Jerarquía de asociación	5	3
CIM Device Specification. DMTF SysDev-WG		
Disipación y energia	10	58
Procesadores	3	19
Controladores y controladores PCI	18	88
Puertos lógicos y grupos de puertos	12	62
Adaptadores de red	7	12
Canales de fibra	7	35
Dispositivos de almacenamiento y extensiones	18	69
Modelo extendido SCC	8	10
Servicios y bibliotecas de almacenamiento	18	118
Dispositivos de interacción con el usuario	8	30
Memoria	4	28
Modems	12	80

Cuadro 6.1: Complejidad de la especificación CIM en términos de número de clases, atributos y relaciones

Especificación CIM	**Clases**	**Atrib.**
CIM Device Specification. DMTF SysDev-WG (Cont.)		
Impresoras, trabajos y servicios de impresión	6	67
Sensores y alarmas	10	43
Grupo de USB y disco	5	16
Jerarquías de dependencias, asociaciones y agregaciones	89	82
CIM Event Specification. DMTF Event/Interop-WG		
Clases indicación	16	39
Filtros y manejadores de indicaciones	3	15
Jerarquía de asociaciones	1	16
CIM Interop Schema. DMTF Interop-WG		
Esquema de interoperabilidad	8	36
Jerarquía de asociaciones	9	2
CIM Metrics Schema. DMTF Application-WG		
Clases de métricas y unidades de trabajo (UoW)	7	31
Jerarquía de asociaciones	12	0
CIM Network Specification. DMTF Networks-WG		
Sistemas y colecciones en red	14	33
Terminaciones de protocolo	12	45
Encaminamiento y reenvío	6	19
Rutas y *Pipes*	6	37
Filtrado y entradas de filtrado	6	36
Memorias intermedias (recursos de red)	1	7
Gestión SNMP	3	7
Encaminamiento OSPF	5	21
Encaminamiento BGP	9	54
Puentes y *switches*	10	47
Redes virtuales (VLAN)	3	1
Calidad de servicio (QoS)	32	71
Jerarquías de dependencias, asociación y agregación	115	46
CIM Physical Specification. DMTF SysDev-WG		
Paquetes físicos, paquetes de almacenamiento, conectores y enlaces	13	73
Componente físico	5	27
Capacidad física y conjuntos de reemplazo	4	12
Estadísticas de almacenamiento	2	18
Jerarquías de asociación y agregación	23	7

Cuadro 6.2: Complejidad de la especificación CIM en términos de número de clases, atributos y relaciones (Cont.)

CIM Policy Specification. DMTF Policy-WG		
Conjuntos de políticas	4	17
Condiciones	4	15
Acciones	3	10
Clases de sistema	1	0
Colección de roles de políticas	3	2
Jerarquías de dependencias y agregaciones	19	5
CIM Support Specification		
Contacto	10	40
Elemento de expresión	7	19
Solución	5	7
Incidente de servicio	5	26
Jerarquías de dependencias y agregaciones	35	0
CIM System Specification. DMTF SysDev-WG		
Ordenador	9	26
Sistemas de ficheros	10	42
Software	0	0
Sistema operativo	1	23
Procesamiento y trabajos	5	33
Servicio de inicialización	4	17
Tiempo	1	16
UNIX	7	59
Recursos del sistema	7	23
Mensajes de log	2	25
Diagnósticos	3	38
Jerarquías de asociaciones, dependencias y agregaciones	61	13
CIM Application Model Specification. DMTF Application-WG		
Características del software instalado	5	24
Comprobación y acciones del software	21	54
Manejadores de dispositivos y software de BIOS	5	15
Jerarquía de asociaciones	30	2
Totales	**1.069**	**2.444**

Cuadro 6.3: Complejidad de la especificación CIM en términos de número de clases, atributos y relaciones (Cont.)

A partir de este conjunto de modelos se ha desarrollado un juego de experimentos consistente en:

- Cada uno de los submodelos especificados en el cuadro 6.1 será formalizado en OWL-FULL, y en lógica descriptiva siguiendo los "mappings" propuestos. En este último caso, se utilizará tanto la sintaxis abstracta descrita en el capítulo 3, como la sintaxis basada en LISP especificada en el apéndice D y que sirve de entrada a motores de inferencia como los propuestos en el cuadro 6.5. Estas formalizaciones se llevarán a cabo utilizado las herramientas visuales de un marco de trabajo desarrollado como parte de esta tesis.

- Una vez formalizados los modelos se validarán las especificaciones obtenidas. Para ello se utilizará la herramienta *vOWLidator* de *BBN Technologies* y el conjunto de herramientas descrito en el cuadro 6.5, principalmente *RACER*, ya que es la que presenta una capacidad expresiva más próxima a la requerida por el "mapping" desarrollado.

- El "mapping" de un modelo CIM a una base de conocimiento terminológico (TBox) propuesto en el punto anterior supone la construcción de una terminología $\mathcal{T}$. Durante la construcción de un modelo CIM es importante descubrir si una nueva clase tiene sentido o, por el contrario, es contradictoria respecto del resto del modelo, en cuyo caso nunca podrá instanciarse de modo consistente. Desde un punto de vista lógico, un nuevo concepto C tiene sentido si existe al menos una interpretación $\mathcal{I}$ que satisface los axiomas de $\mathcal{T}$ y para la que el concepto denota un conjunto no vacío. A esta interpretación se la denomina modelo y se escribe $\mathcal{T} \vDash C$. A esta propiedad del concepto C con respecto a $\mathcal{T}$ se la denomina *satisficibilidad*. Se utilizarán por tanto los servicios de razonamiento ofrecidos por el subsistema DL de la herramienta de modelado (CIMOnt) para verificar que todas las clases creadas son satisfacibles respecto del resto del modelo y que se cumplen las relaciones de generalización/especialización esperadas.

- Una vez comprobada la satisficibilidad de los modelos, se verificará formalmente la consistencia de las bases de conocimiento asertivo obtenidas a partir de las instancias del modelo CIM utilizadas por la solución de gestión distribuida implementada por Microsoft para sus sistemas operativos y que está basada en la iniciativa WBEM. Para ello se utilizará la herramienta (CIMOnt), que permite obtener a partir de estas instancias una base de conocimiento asertivo (ABox) adecuada para el sistema de inferencia RACER. Esta comprobación

Motor de inferencia	**Comentario**
KL-ONE	Pre-DL
KRYPTON (Brachman et al.)	Pre-DL
NIKL (Schmolze et al.)	Pre-DL
PENNI (Robins et al.)	Pre-DL
KL-TWO (Vilain et al.)	Pre-DL
KANDOR (Patel-Schneider)	Pre-DL

Cuadro 6.4: Herramientas de razonamiento no evaluadas

asegurará la imposibilidad de que se extraigan conclusiones incoherentes. Una vez verificada la consistencia de la base de conocimiento asertivo, se podrán realizar pruebas basadas en los siguientes servicios prototípicos:

- Comprobación de instancia, consistente en la comprobación de que una aserción es consecuencia lógica del conocimiento almacenado en la ABox.
- Pertenencia (extensión), consistente en la obtención de todos los individuos que son instancia de un determinado concepto (tanto un concepto básico, como una expresión de concepto).
- Realización (Clasificación de instancias), consistente en la obtención del conjunto de conceptos más específicos en que puede clasificarse un individuo dado[1] $\{C \mid \mathcal{A} \vDash C(a)\}$.

El plan de experimentación contempla el estudio del comportamiento de diferentes motores de razonamiento lógico basado en lógica descriptiva en términos de su capacidad expresiva respecto de Lógica descriptiva $\mathcal{ALE}\mathcal{CNO}\mathcal{Q}^{-}_{\mathcal{H}R^{+}\circ}$ utilizada en el "mapping" CIM-DL propuesto en la tesis, así como de su velocidad de clasificación y deducción. El cuadro 6.5 resume las principales características de estos razonadores lógicos. Se han dejado fuera las herramientas KL-ONE, KRYPTON, KL-TWO y KANDOR, ya que no están directamente basadas en lógica descriptiva, y se ha centrado prioritariamente en la utilización de RACER debido fundamentalmente a su elevada expresividad y su capacidad para manejar bases de conocimiento asertivo.

El diseño experimental incluye también la especificación en ACSL del conjunto de protocolos de especificación recogido en el cuadro 6.6, así como la simulación de los mismos a partir de las especificaciones obtenidas. La mayoría de estos protocolos han sido formalizados por FIPA [FIP02], por lo que el cuadro recoge la versión contemplada. El resto de protocolos aparecen formalizados en el apéndice C.

[1]En este contexto, se entiende por conceptos más específicos aquellos que son mínimos con respecto al orden inducido por el operador de subsunción $\sqsubseteq$.

Motor de inferencia	**Tipo de LD**	**Comentario**
CLASSIC (Borgida et al.)	$\mathcal{ALNF}_{h^-}$	R. casi completo, rápida
LOOM (MacGregor et al.)	$\mathcal{ALCQRIFO}$	R. incompleto, expresiva
BACK	$\mathcal{ALQR}^-$	R. incompleto, expresiva
FLEX (Quantz et al.)	$\mathcal{ALCQRIFO}$	
KRIS (Baader et al.)	$\mathcal{ALCNF}$	R. completo, expresiva
CRACK (Bresciani et al.)	$\mathcal{ALCQRIFO}$	R. completo, expresiva
DLP (Horrocks et al.)	$\mathcal{ALCN}_{reg}$	R. correcto y completo, altamente expresiva
FACT (Horrocks et al.)	$\mathcal{SHIQ}$	R. correcto y completo, altamente expresiva
RACER (Haarslev et al.)	$\mathcal{ALCQHI}_{\mathcal{R}^+}$	R. correcto y completo, Soporte para ABox, altamente expresiva

Cuadro 6.5: Herramientas de razonamiento evaluadas

Protocolo	**Identificador**
Request	SC00026
Query	SC00027
Request	SC00028
Contract Net	SC00029
Iterated Contract Net	SC00030
English Auction	XC00031
Dutch Auction	XC00032
Brokering	SC00033
Recruiting	SC00034
Subscribe	SC00035
Propose	SC00036
Task Compensation	-
Sian Knowledge-based Coordination	-

Cuadro 6.6: Protocolos de interacción evaluados

Por último, el diseño experimental recoge la utilización del *Modelo de Organización* propuesto en la modelización de un caso de aplicación con el fin de demostrar el uso de roles, holónes y dominios de cooperación en la configuración de un modelo de organización adecuado para la gestión de acuerdos sobre nivel de servicios (SLA, del inglés *Service Level Agreement*) en entornos distribuidos. Para ello se han tomado prestadas las ideas de un caso de estudio presentado en [HAN98] y referido a la garantía de la calidad de servicio desde la perspectiva de la seguridad. La sección 5.5 describe el caso de uso.

6.2. Marco de trabajo

Con el fin de demostrar la implementabilidad y verificar la viabilidad de los diferentes modelos propuestos en esta tesis, se han desarrollado una serie de herramientas que, en conjunto, conforman un marco de trabajo para el modelo de arquitectura propuesto sobre el que llevar a cabo los experimentos descritos en dicho diseño.

Se ha desarrollado un entorno integrado de desarrollo (AUML*-IDE) para composición visual de protocolos de interacción. Se trata de un conjunto de herramientas desarrolladas sobre la plataforma *Microsoft Visio*[2] que permiten el modelado visual de protocolos de interacción y la generación automática de las especificaciones ACSL correspondientes a los mismos. Las herramientas desarrolladas permiten también la obtención del interprete asociado a las especificaciones ACSL y, a partir de éste, la simulación del protocolo y la comprobación de diversas propiedades formales del mismo. La figura 6.1 muestra una captura de esta herramienta.

También se ha desarrollado un conjunto de herramientas CASE para modelado visual de ontologías (CIMOnt). Este conjunto de herramientas permiten la obtención de la formalización en lógica descriptiva de los modelos de información CIM creados con *Microsoft Visio*, así como su especificación en OWL. También se incorpora el modelado visual de modelos CIMOnt mediante una notación gráfica propia, más próxima a la lógica descriptiva que a la notación CIM. El componente fundamental de este conjunto de herramientas es el encargado de realizar el "mapping" de los modelos CIM a lógica descriptiva. Este componente permite comprobar la consistencia lógica de los modelos desarrollados (por si mismos y respecto al resto de modelos CIM propuestos por el DMTF) en tiempo de diseño. La figura 6.2 muestra una captura de esta herramienta.

Para el modelado visual de dominios de cooperación (MHolón), se ha creado un conjunto de herramientas, basadas en las anteriores, que permiten la especificación

[2]La elección de esta plataforma ha sido debida a la utilización de la misma por parte del DMTF en la elaboración y publicación de sus estándares CIM.

visual de holarquías, así como su formalización en lógica descriptiva y su expresión en el lenguaje de ontologías OWL. Este último aspecto es especialmente importante, ya que permite verificar en tiempo de diseño la satisficibilidad de los modelos de organización creados. También es de destacar la posibilidad de realizar consultas acerca de los modelos de organización desarrollados tanto en tiempo de diseño como en tiempo de ejecución por parte de los agentes autónomos de gestión (en este último caso, se puede hablar de un *servicio de dominio*). La figura 6.3 muestra una captura de esta herramienta.

Por último, el marco de trabajo dispone de una infraestructura de comunicaciones (COMADA) basada en tecnologías Web y desarrollada sobre las plataformas *Microsoft .NET* y *J2EE-Sun ONE* que ofrece servicios de comunicación a los agentes desarrollados según la arquitectura propuesta. Es de destacar la facilidad aportada por esta infraestructura para la generación automática de proxies en base a técnicas de *reflexión* que permiten a los agentes, a partir de la especificación ACSL de un protocolo de interacción, el seguimiento automático de las conversaciones en que participan. El proxy generado implementa un intérprete basado en la semántica operacional desarrollada para el lenguaje ACSL como parte de esta tesis. La figura 6.4 muestra una simulación de una conversación entre cuatro agentes en la que se siguen dos protocolos de interacción especificados en ACSL. Los cuatro agentes han aprendido estos protocolos de forma dinámica a partir de dicha especificación. La comunicación se realiza a través de HTTP utilizando el formato de mensaje especificado por FIPA.

6.3. Resultados

Un importante resultado novedoso obtenido al aplicar las ideas aportadas en este trabajo de Tesis es que se ha clasificado y verificado la satisficibilidad de la totalidad del modelo CIM propuesto por el DMTF, en su versión 2.7, a partir del "mapping" CIM-$\mathcal{ALE}\mathcal{CNOQ}^{-}_{\mathcal{HR}^{+}\circ}$ presentado en el capítulo 3. Esta versión del modelo está compuesta por 14 submodelos, y presenta un total de 1.069 clases, 2.444 atributos y 1.044 referencias, lo cual da una idea de la magnitud del problema. El cuadro 6.1 desglosa pormenorizadamente estas cantidades. Como resultado de esta clasificación, por primera vez ha sido posible verificar formalmente la consistencia del modelo. En total se han formalizado, clasificado y razonado acerca de las propiedades de un total de 14 modelos cuyas características aparecen resumidas en el cuadro 6.7 y dan una idea de la magnitud del problema:

Otros resultados de gran importancia son los relacionados con la capacidad de razonamiento automático de tipo deductivo que proporcionan los modelos de la ar-

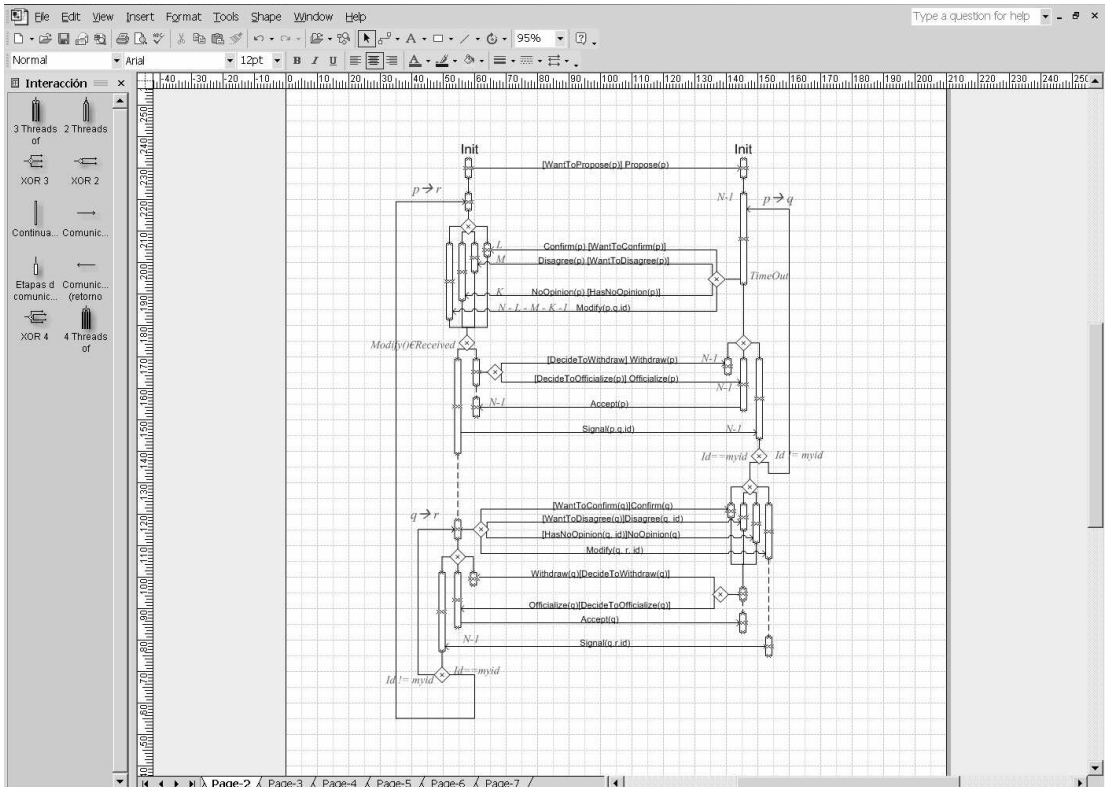

Figura 6.1: Entorno integrado de desarrollo $ACSL - AUML*$

quitectura descrita. Así, la conexión de una herramienta de modelado visual basado en CIM como puede ser *Microsoft Visio* a un motor de razonamiento con soporte para el nivel de expresividad requerido en el "mapping" propuesto (e.g. [HM01]) permite dotar a este tipo de herramientas CASE de capacidades de inferencia lógica acerca de los modelos desarrollados. En este sentido, un diseñador de bases de información de gestión conceptualizadas mediante CIM podría comprobar la consistencia de sus modelos con respecto a los restantes modelos CIM desarrollados por el DMTF y/o por terceras partes en función de los elementos (clases, asociaciones,

Concepto	**Cantidad**
Modelos	14
Submodelos	89
Clases	1.069
Atributos	2.444
Referencias	1.044

Cuadro 6.7: Complejidad del modelo CIM

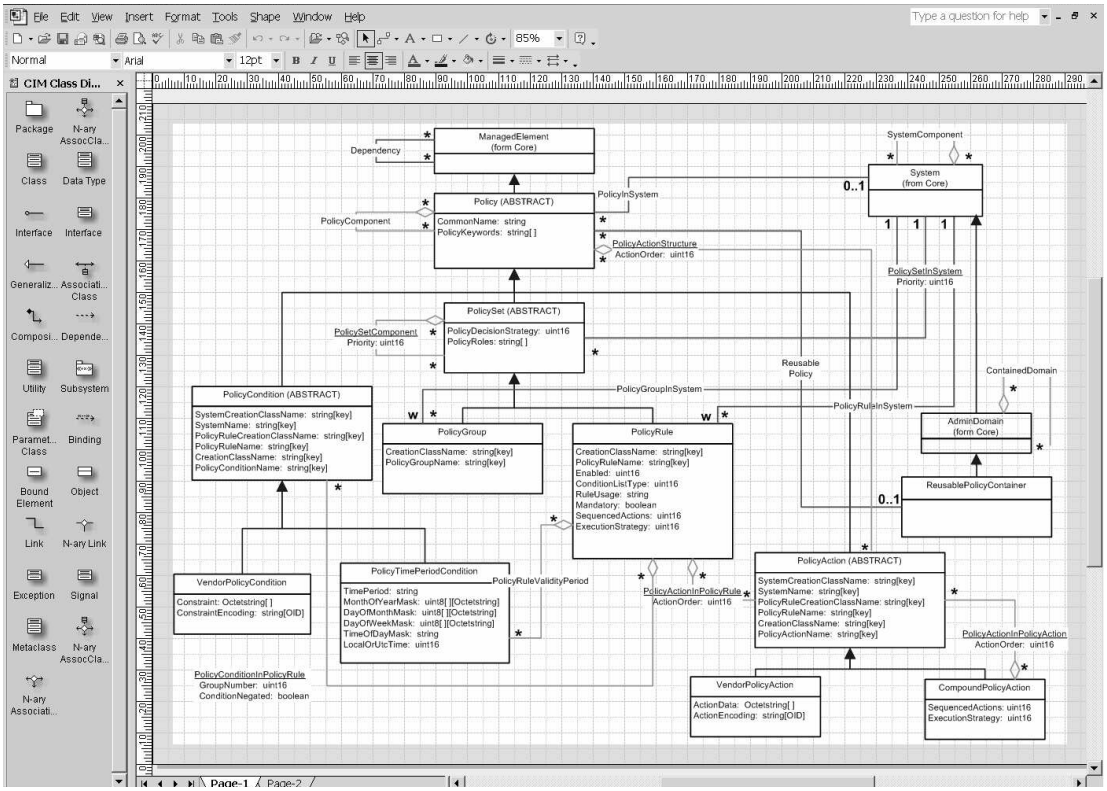

Figura 6.2: Herramienta de modelado visual *CIMOnt*

triggers, indicadores, etc.) de dichos modelos a que haga referencia en sus propios modelos. Estas ideas son válidas y aplicables, por extensión, para otras herramientas de modelado basadas en otros lenguajes de modelado de propósito general como, por ejemplo, UML.

Por último es destacable el hecho de haber podido realizar satisfactoriamente simulaciones de diferentes conversaciones multi-parte y multi-protocolo en las que los agentes aprendieron dinámicamente en tiempo de ejecución los protocolos de interacción a utilizar a partir de especificaciones ACSL obtenidas a través de Internet.

Las propuestas elaboradas en esta tesis son consecuencia directa del trabajo de investigación realizado por el autor en el contexto de diversos proyectos de investigación con diversos organismos públicos y empresas privadas. Entre estos proyectos se encuentra la propuesta "NESMARQ: Arquitectura para Gestión Cooperativa de Redes y Servicios Avanzados de Telecomunicaciones en Entornos Heterogéneos mediante Sociedades de Agentes Autónomos"; que constituye un proyecto de investigación financiado por la Dirección General de Proyectos de Investigación del Ministerio de Ciencia y Tecnología (referencia TIC2001-3451), en el que el autor viene participando desde su propuesta inicial como investigador a tiempo completo.

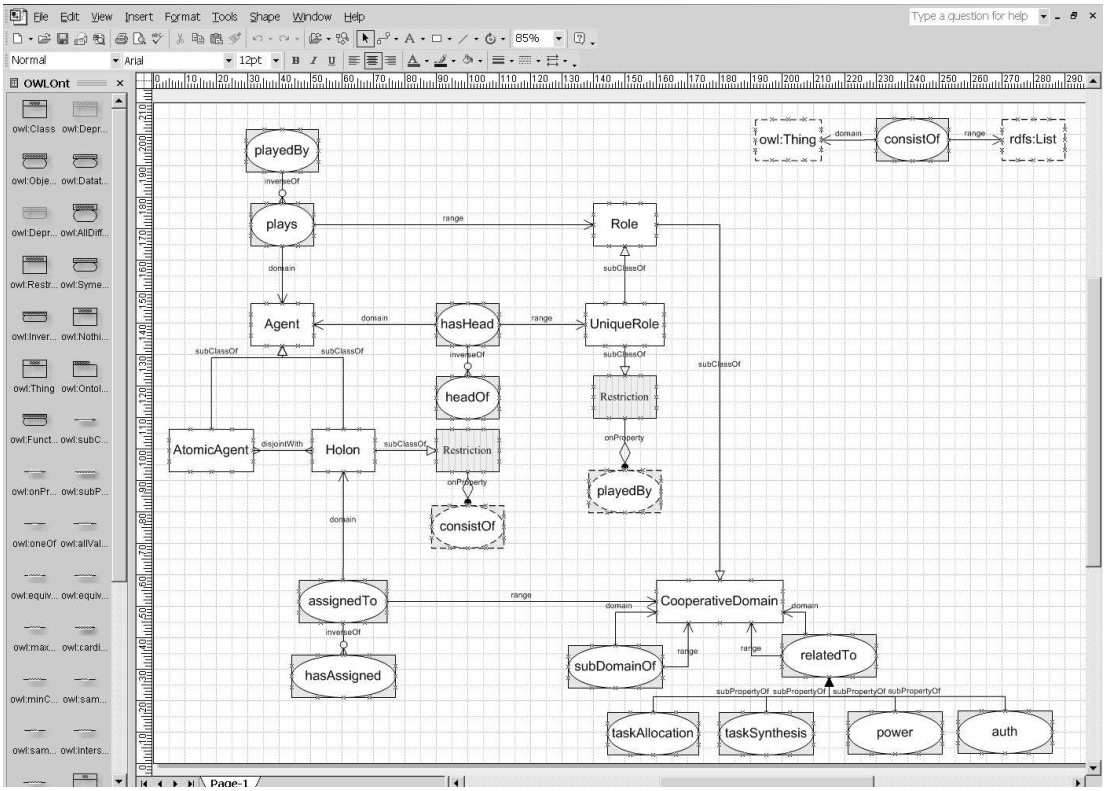

Figura 6.3: Herramienta de modelado visual *MHolon*

Los resultados derivados de estos trabajos de investigación se han producido en el contexto del plan de explotación de resultados científicos asociados a dichos proyectos, e incluyen una serie de publicaciones en revistas y congresos entre los que cabe citar: [SAL+01, SAL03, BHL+00, AFL+01, BFH+01, ALSV02, ABLS02, SLA02, SAL02].

Así, en [SAL03] se presenta el lenguaje de especificación ACSL y se introduce el concepto de *Definición de Interfaz de Agente* (DIA). En [SAL+01] se presenta un modelo de organización para sociedades de agentes y se describen los principales elementos que lo constituyen. En [ALSV02, SLA02] se extienden los modelos de roles existentes, basados en estados mentales (creencias, deseos, metas e intenciones), con nuevos elementos mentalísticos que se agrupan bajo el término paraguas "políticas"(normas, responsabilidades, obligaciones, permisos, prohibiciones, etc.) y se propone la extensión de la arquitectura abstracta de FIPA y su modelo de interoperabilidad. En [SAL02] se formalizan los conceptos presentados en [ALSV02] y [SLA02] y los aspectos relacionados con su dinámica (creación, comunicación, cumplimiento) mediante la lógica modal (lógicas dinámica, ilocucional y deontológica). En [BFH+01, ABLS02] se aplican los conceptos anteriores en la concepción de una

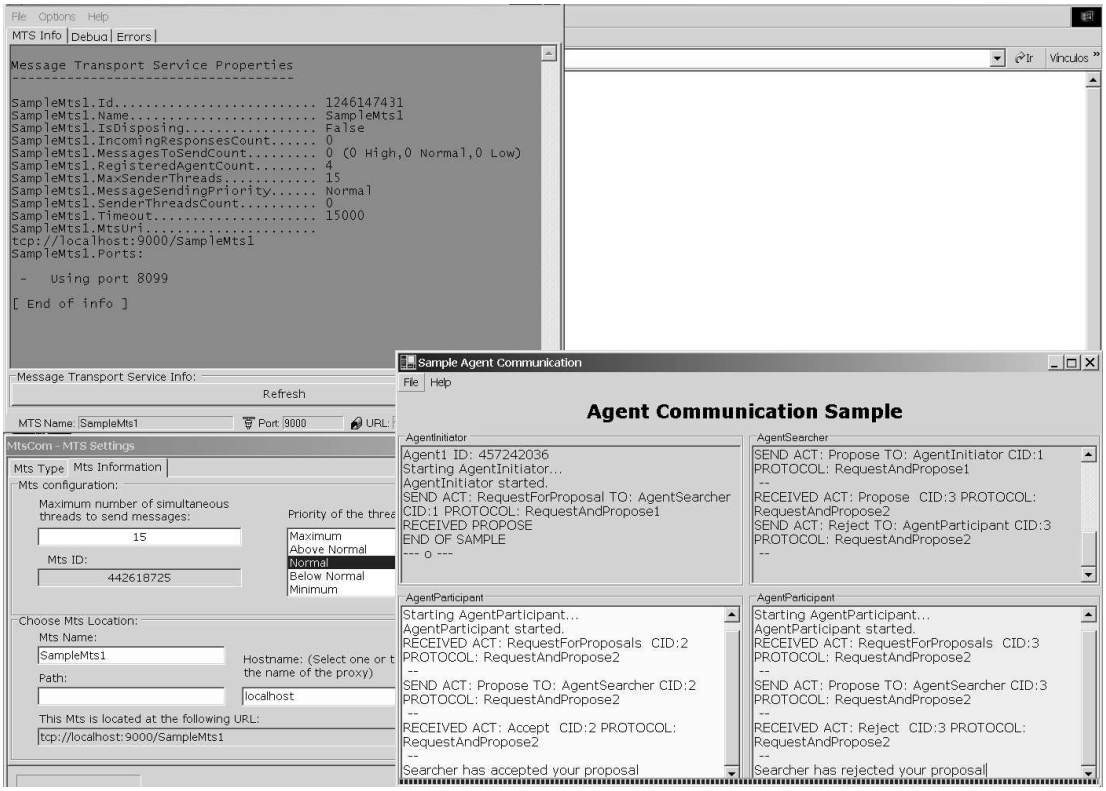

Figura 6.4: Captura de una simulación de conversación multi-protocolo en la plataforma COMADA

arquitectura de gestión de redes que explota el paradigma cooperativo y fuertemente distribuido y la gestión basada en políticas. [AFL+01] presenta la plataforma de agentes subyacente a esta arquitectura. El "mapping" CIM-$\mathcal{ALECNQ}^{-}_{\mathcal{HR}^{+}\circ}$ propuesto en este trabajo de Tesis ha sido enviado recientemente al simposio internacional *IEEE/IFIP Network Operations and Management Symposium* (NOMS 2004).

Capítulo 7

Conclusiones y líneas futuras

Índice General

7.1. Conclusiones 185
7.2. Líneas futuras 190

7.1. Conclusiones

El análisis del espacio de soluciones realizado por el autor, como aportación de este trabajo de Tesis al estado de la cuestión en arquitecturas de gestión, ha mostrado una marcada tendencia en las arquitecturas existentes por incrementar el grado de *automatización* de las soluciones de gestión creadas en base a las mismas, por aumentar su *escalabilidad*, por extender sus capacidades de *cooperación/delegación* y por multiplicar su grado de *interoperabilidad*. Este estudio ha dejado patente también la falta de adecuación de ciertos aspectos intrínsecos a los modelos de información y organización de estas arquitecturas respecto de la creciente complejidad, criticidad y sofisticación asociada a los nuevos servicios telemáticos emergentes, consecuencia directa del profundo proceso de reestructuración que está aconteciendo en el mercado de las telecomunicaciones y las redes de datos. Por último, se han analizado las interdependencias existentes entre estos cuatro factores: *automatización*, *escalabilidad*, *interoperabilidad*, y *cooperación/delegación*, resultando manifiesta la necesidad de considerar simultáneamente todos ellos a la hora de diseñar una nueva arquitectura de gestión. El incremento en el grado de automatización permitido por la arquitectura propuesta se traduce directamente en una mayor escalabilidad y requiere tanto de un mayor grado de interoperabilidad, como de un mejor soporte para la cooperación y la delegación de la funcionalidad de gestión.

A partir de las conclusiones extraídas del análisis del espacio de soluciones, la tesis ha presentado un modelo de arquitectura para gestión cooperativa de sistemas y servicios distribuidos basado en sociedades de agentes autónomos. Los objetivos fundamentales conseguidos con el desarrollo de esta arquitectura han sido permitir la integración de las soluciones de gestión existentes, a la vez que obtener un alto grado interoperabilidad en entornos abiertos, e incrementar el grado de automatización asociado al proceso global de gestión, lo cual junto con la provisión de soporte para delegación por dominios y delegación de macro-tareas, aumenta la escalabilidad de las nuevas soluciones de gestión desarrolladas en base a la misma.

Los modelos propuestos en la nueva arquitectura de gestión suponen las siguientes aportaciones novedosas:

1. Los nuevos requisitos de automatización e interoperabilidad han guiado el desarrollo de un nuevo *Modelo de Información* formal de tipo semántico denominado *CIMOnt*. El modelo está basado en la lógica descriptiva, lo cual permite un mayor grado de automatización en la gestión en base a la utilización de agentes autónomos racionales, capaces de razonar, inferir e integrar de forma dinámica conocimiento y servicios conceptualizados mediante el modelo de información CIM del DMTF y formalizados a nivel semántico mediante lógica descriptiva.

2. El modelo de información propuesto incorpora un "mapping" a nivel de metamodelo del modelo CIM al lenguaje de especificación de ontologías OWL, que supone un significativo avance en el área de la representación y el intercambio basado en XML de modelos y meta-información. Las propuestas existentes están basadas en lenguajes XML sin soporte semántico de tipo formal. Sin embargo, la posibilidad de utilizar la representación XML de la formalización en lógica descriptiva de una conceptualización CIM como sintaxis concreta de transferencia, añade al modelo de intercambio propuesto nuevas e importantes posibilidades de razonamiento automático acerca tanto de los modelos conceptuales manejados, como de la propia información de gestión (y por tanto, acerca del conocimiento asertivo).

3. A nivel de interacción, el modelo de información propuesto ha aportado un nuevo lenguaje de especificación formal de conversaciones entre agentes (ACSL, del inglés *Agent Conversation Specification Language*), que permite la compartición e interpretación automática de definiciones de interfaces de agentes de tipo conversacional basadas en actos ilocucionales. El lenguaje propuesto introduce nuevas capacidades en la especificación de protocolos de interacción,

como son el soporte para excepciones y protocolos de compensación adecuados al carácter autónomo de los agentes de gestión y que facilitan el modelado de contra-propuestas que satisfagan parcialmente la propuesta original y la cancelación de compromisos, la composición por entrelazado y anidamiento de protocolos de interacción, la especificación de elementos de correlación asociados a múltiples protocolos compuestos por entrelazado, la especificación de aspectos semánticos mediante conjuntos de parámetros, etc.

4. Asociada al lenguaje de especificación ACSL, se ha elaborado también una semántica operacional que facilitará la labor de verificación de propiedades formales asociadas a los protocolos de interacción especificados en base al mismo, tales como la alcanzabilidad de estados conversacionales, la finalización del protocolo en tiempo finito o la acotación del número de mensajes intercambiados. El lenguaje ACSL desarrollado está orientado a actos ilocucionales y lenguajes ACL y guarda estrecha relación con la notación gráfica A-UML, lo cual facilitará su aceptación por parte de la comunidad internacional de DAI.

5. Se ha desarrollado también un novedoso *Modelo de Organización* basado en el concepto de holón y orientado a roles cuyas principales características están alineadas con las demandadas por los servicios distribuidos emergentes. Entre estas características se incluyen la ausencia de control central, capacidades de restructuración dinámica, capacidades de cooperación, negociación y delegación automática, así como facilidades de adaptación a diferentes culturas organizativas. El modelo incluye un submodelo normativo que se ha mostrado adecuado al carácter autónomo de los holones de gestión y que está fundamentado en las lógicas modales deontológica y de acción.

6. Con el fin de demostrar la implementabilidad y verificar la viabilidad de los diferentes modelos propuestos en esta tesis, se ha elaborado un diseño experimental y se han desarrollado una serie de herramientas que, en conjunto, conforman un marco de trabajo para el modelo de arquitectura propuesto sobre el que llevar a cabo los experimentos descritos en dicho diseño. Este marco de trabajo está compuesto por:

   - Un entorno integrado de desarrollo (AUML*-IDE) para composición visual de protocolos de interacción. Se trata de un conjunto de herramientas desarrolladas sobre la plataforma *Microsoft Visio*[1] que permiten el modelado visual de protocolos de interacción y la generación automática de las

[1] La elección de esta plataforma ha sido debida a la utilización de la misma por parte del DMTF en la elaboración y publicación de sus estándares CIM.

especificaciones ACSL correspondientes a los mismos. Las herramientas desarrolladas permiten también la obtención del interprete asociado a las especificaciones ACSL y, a partir de éste, la simulación del protocolo y la comprobación de diversas propiedades formales del mismo.

- Herramientas CASE para modelado visual de ontologías (CIMOnt). Este conjunto de herramientas permiten la obtención de la formalización en lógica descriptiva de los modelos de información CIM creados con *Microsoft Visio*, así como su especificación en OWL. También se incorpora el modelado visual de modelos CIMOnt mediante una notación gráfica propia, más próxima a la lógica descriptiva que a la notación CIM. El componente fundamental de este conjunto de herramientas es el encargado de realizar el "mapping" de los modelos CIM a lógica descriptiva. Este componente permite comprobar la consistencia lógica de los modelos desarrollados (por si mismos y respecto al resto de modelos CIM propuestos por el DMTF) en tiempo de diseño.

- Herramientas para modelado visual de dominios de cooperación (MHolón). Se trata de un conjunto de herramientas, basadas en las anteriores, que permiten la especificación visual de holarquías, así como su formalización en lógica descriptiva y su expresión en el lenguaje de ontologías OWL. Este último aspecto es especialmente importante, ya que permite verificar en tiempo de diseño la satisficibilidad de los modelos de organización creados. También es de destacar la posibilidad de realizar consultas acerca de los modelos de organización desarrollados tanto en tiempo de diseño como en tiempo de ejecución por parte de los agentes autónomos de gestión (en este último caso, se puede hablar de un *servicio de dominio*).

- Una infraestructura de comunicaciones basada en tecnologías Web y desarrollada sobre las plataformas *Microsoft .NET* y *J2EE-Sun ONE* que ofrece servicios de comunicación a los agentes desarrollados según la arquitectura propuesta. Es de destacar la facilidad aportada por esta infraestructura para la generación automática de proxies en base a técnicas de *reflexión* que permiten a los agentes, a partir de la especificación ACSL de un protocolo de interacción, el seguimiento automático de las conversaciones en que participan. El proxy generado implementa un intérprete basado en la semántica operacional desarrollada para el lenguaje ACSL como parte de esta tesis.

Un importante resultado novedoso obtenido al aplicar las ideas aportadas en este trabajo de Tesis es que se ha clasificado la totalidad del modelo CIM propuesto por el DMTF, en su versión 2.7, a partir del "mapping" CIM-$\mathcal{AL}\varepsilon\mathcal{CNOQ}^{-}_{\mathcal{H}R^{+\circ}}$ presentado en el capítulo 3. Esta versión del modelo está compuesta por 14 submodelos, y presenta un total de 1.069 clases, 2.444 atributos y 1.044 referencias, lo cual da una idea de la magnitud del problema. El cuadro A.1 desglosa pormenorizadamente estas cantidades. Como resultado de esta clasificación, por primera vez ha sido posible verificar formalmente la consistencia del modelo.

Otros resultados de gran importancia son los relacionados con la capacidad de razonamiento automático de tipo deductivo que proporcionan los modelos de la arquitectura descrita. Así, la conexión de una herramienta de modelado visual basado en CIM como puede ser *Microsoft Visio* a un motor de razonamiento con soporte para el nivel de expresividad requerido en el "mapping" propuesto (e.g. [HM01]) permite dotar a este tipo de herramientas CASE de capacidades de inferencia lógica acerca de los modelos desarrollados. En este sentido, un diseñador de bases de información de gestión conceptualizadas mediante CIM podría comprobar la consistencia de sus modelos con respecto a los restantes modelos CIM desarrollados por el DMTF y/o por terceras partes en función de los elementos (clases, asociaciones, triggers, indicadores, etc.) de dichos modelos a que haga referencia en sus propios modelos. Estas ideas son válidas y aplicables, por extensión, para otras herramientas de modelado basadas en otros lenguajes de modelado de propósito general como, por ejemplo, UML.

Por último es destacable el hecho de haber podido realizar satisfactoriamente simulaciones de diferentes conversaciones multi-parte y multi-protocolo en las que los agentes aprendieron dinámicamente en tiempo de ejecución los protocolos de interacción a utilizar a partir de especificaciones ACSL obtenidas a través de Internet.

Las propuestas elaboradas en esta tesis son consecuencia directa del trabajo de investigación realizado por el autor en el contexto de diversos proyectos de investigación con diversos organismos públicos y empresas privadas. Entre estos proyectos se encuentra la propuesta "NESMARQ: Arquitectura para Gestión Cooperativa de Redes y Servicios Avanzados de Telecomunicaciones en Entornos Heterogéneos mediante Sociedades de Agentes Autónomos"; que constituye un proyecto de investigación financiado por la Dirección General de Proyectos de Investigación del Ministerio de Ciencia y Tecnología (referencia TIC2001-3451), en el que el autor viene participando desde su propuesta inicial como investigador a tiempo completo.

Los resultados derivados de estos trabajos de investigación se han producido en el contexto del plan de explotación de resultados científicos asociados a dichos

proyectos, e incluyen una serie de publicaciones en revistas y congresos entre los que cabe citar: [SAL+01, SAL03, BHL+00, AFL+01, BFH+01, ALSV02, ABLS02, SLA02, SAL02].

7.2. Líneas futuras

Si bien este trabajo de Tesis ha sentado las bases de modelado para futuras arquitecturas de gestión basadas en el paradigma cooperativo y fuertemente distribuido, el problema es de considerable magnitud y aún queda mucho trabajo por hacer en esta nueva dirección.

Por una parte, la aplicación práctica del Modelo de Información propuesto se basa en la utilización de motores de razonamiento automático existentes para lógica descriptiva, como [HM01], [FACT] o [DLP]. Se hace imprescindible entonces el estudio de la complejidad computacional del proceso de razonamiento acerca de modelos conceptualizados en CIM llevado a cabo por estos motores, así como la propuesta de mecanismos específicos orientados a mejorar dicha complejidad.

En este mismo sentido, se abre un gran abanico de posibilidades para el desarrollo de herramientas de modelado visual con soporte para razonamiento automático acerca de los modelos creados mediante las mismas, así como la adaptación de las existentes.

Por otra parte, la formalización llevada a cabo en el área de la especificación de protocolos conversacionales introduce nuevas posibilidades en el diseño de los mismos. Destacamos las relacionadas con el modelado composicional de los mismos y las relacionadas con la verificación de propiedades y su validación. En este sentido, la semántica operacional desarrollada para el lenguaje ACSL facilitará enormemente la elaboración de herramientas de análisis formal acerca de propiedades dinámicas del lenguaje de especificación, tales como la terminación de un protocolo de interacción en tiempo finito, la alcanzabilidad de estados, determinismo en la ejecución de un protocolo, etc.

El análisis de los diferentes motores de razonamiento existentes en el mercado respecto del "mapping" CIM-$\mathcal{ALECNOQ}^{-}_{\mathcal{HR}^{+\circ}}$ propuesto ha revelado en todos ellos, en mayor o menor medida, un grado de expresividad insuficiente. Este hecho debería servir de acicate para investigar nuevas técnicas de implementación y optimización plausibles para la lógica propuesta.

Finalmente, a partir de los modelos de información y organización presentados en esta Tesis, deberán desarrollarse especificaciones concretas adaptadas a las necesidades particulares de cada organización. El hecho de que el modelo de información desarrollado esté fundamentado en las ontologías como lenguaje de representación de

conocimiento, facilitará enormemente la reutilización de estas especificaciones. Las herramientas desarrolladas como parte del marco de trabajo para la arquitectura propuesta constituyen un punto de partida conveniente.

Bibliografía

[ABLS02] F. Alonso, N. Barcia, G. López, and J. Soriano. Enabling policy-based networking by means of intelligent agent societies. In *Proceedings of the 6th World Multiconference on Systemics, Cybernetics and Informatics (SCI-02)*, Orlando, Florida, USA, 2002.

[AFL+01] F. Alonso, F. Fernández, G. López, P. Rojas, J. Soriano, and J. Vélez. COMADA: Una plataforma para el desarrollo de servicios inteligentes en internet basados en sociedades de agentes. In *Simposio en Informatica y Telecomunicaciones (SIT-01)*, A Coruña, España, 2001.

[AJP97] F. Alonso, N. Juristo, and J. Pazos. Software engineering and knowldedge engineering: Towards a common life cycle. *The Journal of Software and Systems*, 1997.

[AK93] ARPA-KSF, July 1993. Specification of the KQML Agent-Communication Language, ARPA Knowledge Sharing Initiative, External Interfaces Working Group.

[ALSV02] F. Alonso, G. López, J. Soriano, and J. Vélez. Extending the FIPA interoperability model to deal with agent social issues. In *Proceedings of the 6th World Multiconference on Systemics, Cybernetics and Informatics (SCI-02)*, Orlando, Florida, USA, 2002.

[AR96] P. E. Agre and S. J. Rosenschein, editors. *Computational Theories of Interaction and Agency.* MIT Press, Cambridge, MA, 1996.

[Ari73] Aristoteles. *Obras.* Aguilar, Madrid, Spain, 1973.

[Aus62] J. L. Austin. *How to Do Things with Words.* Harvard University Press, Cambridge, MA, 1962.

[BA90] M. Ben-Ari. *Principles of Concurrent and Distributed Programming.* Prentice-Hall, Englewood Cliffs, NJ, 1990.

[BDBW97] J. M. Bradshaw, S. Dutfeld, P. Benoit, and J. D. Woolley. *Kaos: Towards an Industrial-Strength Open Agent Architecture.* AAAI-MIT Press, 1997.

[BDKJT97] F. M. T. Brazier, B. M. Dunin-Keplicz, N. R. Jennings, and J. Traum. Desire: Modelling multi-agent systems in a compositional formal framework. *International Journal of Cooperative Systems*, 6(1):67–94, 1997.

[BdRV01] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic.* Cambridge University Press, Cambridge, MA, 2001.

[BF95] M. Barbuceanu and M. S. Fox. Cool: A language for describing coordination in multiagent systems. In Victor Lesser and Les Gasser, editors, *Proceedings of the First International Conference oil Multi-Agent Systems (ICMAS-95)*, pages 17–24, San Francisco, CA, USA, 1995. AAAI Press.

[BFH+01] N. Barcia, C. Fernández, F. Hernández, G. López, J. Soriano, and J. Yágüez. NESMARQ: Una arquitectura para gestión de redes y servicios avanzados de telecomunicaciones en entornos heterogineos basada en sociedades de agentes. In *Simposio en Informática y Telecomunicaciones (SIT-01)*, A Coruña, España, 2001.

[BG88] A. H. Bond and L. Gasser, editors. *Readings in Distributed Artificial Intelligence*. Morgan Kaufmann, San Mateo, CA, 1988.

[BGM98] M. Barbuceanu, T. Gray, and S. Mankovski. Coordinating with obligations. In Katia P. Sycara and Michael Wooldridge, editors, *Proceedings of the 2nd International Conference on Autonomous Agents (Agents'98)*, pages 62–69, New York, 9–13, 1998. ACM Press.

[BHL+00] N. Barcia, F. Hernández, G. López, J. Soriano, and J. Yágüez. Un nuevo sistema de gestión remota para entornos operativos en internet. In *Simposio Español de Informática Distribuida (SEID-00)*, Ourense, España, 2000.

[BMO] B. Bauer, J. P. Müller, and J. Odell. Agent UML: A formalism for specifying multiagent software systems. In P. Ciancarini and M. Wooldridge, editors, *Proceedings of the First International Workshop on Agent-Oriented Software Engineering (AOSE00)*, Limerick, Ireland. Springer Verlag.

[Boo94] G. Booch. *Object-Oriented Analysis and Design*. Addison-Wesley, Reading, MA, 1994.

[BPC00] C. Bohoris, G. Pavlou, and H. Cruickshank. Using mobile agents for network performance management. In *Proceedings of the IFIP/IEEE Network Operations and Management Symposium (NOMS'00)*, Hawaii, USA, April 2000. IEEE, IEEE Press.

[Bra97] J. M. Bradshaw, editor. *Software Agents*. AAAI-MIT Press, 1997.

[BS97] G. Blair and J.-B. Stefani. *Open Distributed Processing and Multimedia*. Addison Wesley Longman, 1997.

[BT79] B. J. Biddle and E. J. Thomas. *Role theory: Concepts and Research*. Robert E. Krieger Publishing Company, New York, USA, 1979.

[Bur96] B. Burmeister. Models and methodologies for agent-oriented analysis and design. In K. Fischer, editor, *Working notes of the KI'96 Workshop on Agent-Oriented Programming and Distributed Systems*, 1996.

[BW98] U. Blumenthal and B. Wijnen. RFC 2274: User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3), January 1998.

[CC96] P. Ciancarini and C.Hankin, editors. *Coordination Languages and Models - Proceedings of Coordination 96*, volume 1957 of *LNCS*, Heidelberg, Berlin, 1996. Springer Verlag.

[CCF+99] R. Cost, Y. Chen, T. Finin, Y. Labrou, and Y. Peng. Modeling agent conversations with colored petri nets. In *Working Notes of the Workshop on Specifying and Implementing Conversation Policies*, pages 59–66, Seattle, Washington, May 1999.

[CFSD90] J. D. Case, M. Fedor, M. L. Schoffstall, and C. Davin. RFC 1157: Simple network management protocol (SNMP), May 1990.

[CHPW98] J. Case, D. Harrington, R. Presuhn, and B. Wijnen. RFC 2272: Message processing and dispatching for the Simple Network Management Protocol (SNMP), January 1998.

[CMRW93] J. Case, K. McCloghrie, M. Rose, and S. Waldbusser. RFC 1451: Manager-to-manager management information base, April 1993.

[CvHH+01] D. Connolly, F. van Harmelen, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein. DAML+OIL reference description. Technical report, DARPA Knowledge Sharing Force, 2001.

[Dam02] N.C. Damianou. *A Policy Framework for Management of Distributed Systems*. PhD thesis, Imperial College of Science, Technology & Medicine, University of London, Department of Computing, 2002.

[DC96] Y. Demaeau and A. C. Rocha Costa. Populations and organizations in open multi-agent systems. In *Proceedings of the 1st National Symposium on Parallel and Distributed AI*, 1996.

[DdS02] Y. Demaeau and J. L. da Silva. Constraining autonomy through norms. In M. Wooldridge et al., editor, *Proceedings of the Autonomous Agents and Multi-Agent Systems (AAMAS)*, Bologna, Italy, July 2002.

[Dec00] S. Decker. The semantic web: The roles of XML and RDF. *IEEE Internet Computing*, 4(5):63–74, 2000.

[Dig99] F. Dignum. Autonomous agents with norms. *Artificial Intelligence and Law*, 7:69–79, 1999.

[Dig00a] F. Dignum. Agent communication and cooperative information agents. In *Cooperative Information Agents*, pages 191–207, 2000.

[Dig00b] F. Dignum. Flbc: From messages to protocols. In F. Dignum and C. Sierra, editors, *European Perspective on Agent Mediated Electronic Commerce*, Heidelberg, Berlin, 2000. Springer Verlag.

[DJM+00] O. Dieste, N. Juristo, A. M. Moreno, J. Pazos, and A. Sierra. *Handbook of Software Engineering and Knowledge Engineering*, volume 1, chapter Conceptual Modelling in Software Engineering and Knowledge Engineering: Concepts, Techniques and Trends. World Scientific Publishing Company, 2000.

[DMT03] Web-based enterprise management (WBEM). Technical report, Distributed Management Task Force, 2003.

[DSS93] R. Davis, H. Shrobe, and P. Szolovits. What is a knowledge representation? *AI Magazine*, pages 17–33, Spring 1993.

[DUB03] Dublin core metadata element set. Recommendation, Februrary 2003.

[Dur94] E. Durkheim. Les règles de la méthode sociologique. *Revue Philosophique*, 37-38:465–498, 1894.

[DW95] F. Dignum and H. Weigand. Communication and deontic logic. In R. Wieringa and R. Feenstra, editors, *Information Systems, Correctness and Reusability*, pages 242–260, Singapore, 1995. World Scientific.

[ea99] M. Georgeff et al. *The Belief-Desire-Intention Model of Agency*, volume 1555 of *LNCS*, pages 1–10. Springer Verlag, Heidelberg, Berlin, 1999.

[ea01] N. Yoshioka et al. Safety and security in mobile agents. In P. Ciancarini and M. Wooldridge, editors, *Agent-Oriented Software Engineering - Proceedings of the 1st International Workshop AOSE-2000*, number 1957 in LNCS, pages 223–235, Heidelberg, Berlin, 2001. Springer Verlag.

[EBHM00] R. van Eijk, F. de Boer, W. van der Hoek, and J-Ch. Meyer. *Issues in Agent Communication*, chapter Operational Semantics for Agent Communication Languages, pages 80–95. Number 1916 in Lecture Notes in Computer Science. Springer-Verlag, Heidelberg, Berlin, 2000.

[Eva97] A. S. Evans. Foundations of the Unified Modeling Language. In D. Duke and A. S. Evans, editors, *Proceedings of the 2nd Northen Formal Methods Workshop*, LNCS, pages 75–81, Heidelberg, Germany, 1997. Springer Verlag.

[Eva98] A. S. Evans. Reasoning with UML class diagrams. In *Proceedings of the 2nd Workshop on Industrial Strength Formal Specification Techniques*. IEEE Computer Society Press, 1998.

[Fei95] S. Feit. *A Guide to Network Management*. McGraw-Hill, 1995.

[Fer99] J. Ferber. *Multi-Agent Systems*. Addison-Wesley, Reading, MA, 1999.

[FFMM94] T. Finin, R. Fritzson, D. McKay, and R. McEntire. KQML as an Agent Communication Language. In N. Adam, B. Bhargava, and Y. Yesha, editors, *Proceedings of the 3rd International Conference on Information and Knowledge Management (CIKM'94)*, pages 456–463, Gaithersburg, MD, USA, 1994. ACM Press.

[FG98] J. Ferber and O. Gutknecht. A meta-model for the analysis and design of organizations in multi-agent systems. In *Proceedings of the Third International Conference on Multi-Agent Systems (ICMAS98)*, IEEE CS Press, pages 128–135, Paris, France, 1998. IEEE, IEEE.

[FGPPP99] M. Fernández, A. Gómez-Pérez, J. Pazos, and A. Pazos. Building a chemical ontology using methontology and the ontology design environment. *IEEE Intelligent Systems*, 14(1):37–46, Nov 1999.

[FHH+00] D. Fensel, I. Horrocks, F. van Harmelen, S. Decker, M. Erdmann, and M. C. A. Klein. OIL in a nutshell. In *Knowledge Acquisition, Modeling and Management*, pages 1–16, 2000.

[FIP02] FIPA, October 2002. FIPA 2002 Specification Part 2: Agent Communication Language. Technical Report. FIPA - Foundation for Intelligent Physical Agents.

[FK85] R. Fikes and T. Kehler. The role of frame-based representation in reasoning. *Communications of the ACM*, 28(9):904–902, September 1985.

[FM01] D. Fensel and M. A. Musen. The semantic web. *IEEE Intelligent Systems*, 16(2):24–25, 2001.

[FPV98] A. Fuggetta, G. P. Picco, and G. Vigna. Understanding code mobility. *IEEE Transactions on Software Engineering, 1998*, 1998.

[FW94] M. Fisher and M. Wooldridge. Specifying and executing protocols for cooperative action. In *International Working Conference on Cooperating Knowledge-Based Systems*, Keele, 1994.

[GB99] A. Galan and A. Baker. Multi-agent communication in JAFMAS. In *Working Notes of the Workshop on Specifying and Implementing Conversation Policies*, pages 67–70, Seattle, Washington, May 1999.

[GF92] M. Genesereth and R. Fikes. Knowledge interchange format, version 3.0 reference manual. Technical Report, Computer Science Department, Stanford University, 1992.

[GH99] L. Gasser and M. Huhns, editors. *Distributed Artificial Intelligence, Volume II*. Morgan Kaufman, San Mateo, CA, 1999.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, New York, 1979.

[GJMP97] A. Gómez, N. Juristo, C. Montes, and J. Pazos. *Ingeniería del Conocimiento*. Editorial Centro de Estudios Ramón Areces, Madrid, España, 1997.

[GK97] M. R. Genesereth and S. P. Ketchpel. Software agents. *Communications of the ACM*, 37(7), 1997.

[Góm97] A. Gómez. *Knowledge Sharing and Reuse*. CRC Press, 1997.

[GPG00] D. Griffin, G. Pavlou, and P. Georgatsos. Providing customisable network management services through mobile agents. In *Proceedings of the 7th International Conference on Intelligence in Services and Networks (IS&N00)*, Athens, Greece, 2000.

[Gru91] T. R. Gruber. The role of common ontology in achieving sharable, reusable knowledge bases. In R. Fikes and E. Sandewall, editors, *Proceedings of the Knowledge Representation and Reasoning*, San Mateo, CA, 1991. Morgan Kaufmann.

[Gru92] T. R. Gruber. Ontolingua: A mechanism to support portable ontologies, 1992.

[Gru93] T. R. Gruber. Towards Principles for the Design of Ontologies Used for Knowledge Sharing. In N. Guarino and R. Poli, editors, *Formal Ontology in Conceptual Analysis and Knowledge Representation*, Deventer, The Netherlands, 1993. Kluwer Academic Publishers.

[Gua98] N. Guarino. Formal ontology and information systems. In N. Guarino, editor, *Proceedings of the 1st International Conference on Formal Ontologies in Information Systems, FOIS'98*, pages 3–15, Trento, Italy, Jun 1998.

[GY95] G. Goldszmidt and Y. Yemini. Distributed management by delegation. In *Proceedings of the 15th International Conference on Distributed Computing Systems (ICDCS'95)*, Vancouver, Canada, May 1995. IEEE, IEEE Press.

[GY98] G. Goldszmidt and Y. Yemini. Delegated agents for network management. *IEEE Transacrtions on Software Engineering*, 36(3), March 1998.

[Had96] A. Haddadi. Communication and cooperation in agent systems: A pragmatic theory. *Lecture Notes in Computer Science*, 1056, 1996.

[HAN98] H. Hegering, S. Abeck, and B. Neumair. *Integrated Management of Networked Systems: Concepts, Architectures and their Operational Application.* Series in Networking. Morgan Kaufmann, 1998.

[Har87] D. Harel. Statecharts: A visual formalism for complex systems. *Science of Computer Programming*, 8(3):231–274, June 1987.

[HB99] A. L. G. Hayzelden and J. Bigham, editors. *Software Agents for Future Communication Systems.* Springer-Verlag, Heidelberg, Berlin, 1999.

[HB01] A. L. G. Hayzelden and R. A. Bourne, editors. *Agent Technology for Communication Infrastructures.* John Wiley and Sons, LTD, 2001.

[Hen90] M. Hennessy. *The Semantics of Programming Languages: An Introduction Using Structured Operational Semantics.* Wiley, 1990.

[HGT+94] T. Hasegawa, L. Gou, S. Tamura, P. B. Luh, and J. M. Oblak. Holonic planning and scheduling architecture for manufacturing. In *Proceedings of the International Working Conference on Cooperating Knowledge Based Systems*, pages 125–139, Keele, U.K., June 1994.

[HK85] P. Harmon and D. King. *Expert Systems.* John Wiley and Sons, Inc., USA, 1985.

[HK86] P. Harmon and D. King. *Expert Systems: Artificial Intelligence in Business.* John Wiley and Sons, Inc., New York, NY, 1986.

[HM01] V. Haarslev and R. Müller. RACER system description. In *Proceedings of the IJCAR 2001*, number 2083 in LNAI, pages 701–705, Heidelberg, Berlin, 2001. Springer Verlag.

[Hol91] G. Holzmann. *Design and Validation of Computer Protocols.* Prentice Hall International, Hemel Hempstead, UK, 1991.

[Hor02] I. Horrocks. Daml+oil: A description logic for the semantic web. Technical report, IEEE Bulletin of the Technical Committee on Data Engineering, 2002.

[HPW98] D. Harrington, R. Presuhn, and B. Wijnen. RFC 2271: An architecture for describing SNMP management frameworks, January 1998.

[HRWL83] F. Hayes-Roth, D. A. Waterman, and D. B. Lenat, editors. *Building Expert Systems.* Addison-Wesley, USA, 1983.

[HS97] M. Huhns and M. P. Singh, editors. *Readings in Agents*. Morgan Kaufmann, San Mateo, CA, USA, 1997.

[HST99] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In Springer Verlag, editor, *LPAR'99*, number 1705 in LNCS, pages 161–180, Heidelberg, Berlin, 1999. Springer Verlag.

[Huh01] M. N. Huhns. Interaction-oriented programming. In P. Ciancarini and M. Wooldridge, editors, *Agent Oriented Software Engineering - Proceedings of the 1st International Workshop AOSE-2000*, volume 1957 of *LNCS*, Heidelberg, Berlin, 2001. Springer Verlag.

[IET97] IETF. Augmented BNF for Syntax Specification: ABNF, November 1997.

[IGG99] C. Iglesias, M. Garijo, and J. Gonzalez. A survey of agent-oriented methodologies. In Jörg Müller, Munindar P. Singh, and Anand S. Rao, editors, *Proceedings of the 5th International Workshop on Intelligent Agents V : Agent Theories, Architectures, and Languages (ATAL-98)*, volume 1555, pages 317–330. Springer-Verlag: Heidelberg, Germany, 1999.

[IL00] M. D'Inverno and M. Luck. Formalising the contract net as a goal-directed system. In F. Dignum and C. Sierra, editors, *Workshop MAAMAW-96*, Heidelberg, Berlin, 2000. Springer Verlag.

[IL02] M. D'Inverno and M. Luck, editors. *Understanding Agent Systems*. Springer-Verlag, 2002.

[ISO95] Open distributed management architecture. working draft 3. Working Draft N1851, International Organisation for Standardisation, 1995.

[IT91a] ITU-T. Data communication networks: Open systems interconnection (OSI); management. common management information protocol specification for CCITT applications. ITU-T recommendation X.711. *ITU-T, Geneva, Switzerland*, 1991.

[IT91b] ITU-T. Data communication networks: Open systems interconnection (OSI); management. common management information service definition for CCITT applications. ITU-T recommendation X.710. *ITU-T, Geneva, Switzerland*, 1991.

[IT92a] ITU-T. Data communication networks-information technology: Open systems interconnection-systems management overview. ITU-T recommendation X.701. *ITU-T, Geneva, Switzerland*, 1992.

[IT92b] ITU-T. Data communication networks-information technology: Open systems interconnection (OSI); structure of information management: Guidelines for the definition of managed objects. ITU-T recommendation X.722. *ITU-T, Geneva, Switzerland*, 1992.

[IT92c] ITU-T. Information technology, open systems inteconnection, system management, part 19: Management domain and management policy management functions. recommendation X.749 (ISO DIS 10164-19). *ITU-T, Geneva, Switzerland*, 1992.

[IT92d] ITU-T. TMN management functions. ITU-T recommendation M.3400. *ITU-T, Geneva, Switzerland*, 1992.

[IT95] ITU-T. Case data interchange format (CDIF) and ITU-T recommendations x.901-904 / ISO/IEC 10746. Technical report, ITU-T, 1995.

[IT97a] ITU-T. Recommendation Q.811, lower layer protocol profiles for the Q3 and X interfaces. *ITU-T, Geneva, Switzerland*, 1997.

[IT97b] ITU-T. Recommendation Q.812, upper layer protocol profiles for the Q3 and X interfaces. *ITU-T, Geneva, Switzerland*, 1997.

[IT00] ITU-T. Recommendation m.3010, principles of telecommunications management network (TMN). Specification Circ. 229/258 COM 4-151, ITU-T, February 2000.

[Jen00] N. R. Jennings. On agent-based software engineering. *Artificial Intelligence*, 177(2):277–296, 2000.

[JOB00] H. V. D. Parunak J. Odell and B. Bauer. Representing agent interaction protocols in uml. In P. Ciancarini and M. Wooldridge, editors, *Proceedings of the First International Workshop on Agent-Oriented Software Engineering*, Limerick, Ireland, 2000.

[Ken99] E. A. Kendall. Role modelling for agent system analysis, design and implementation. In *1st International Symposium on Agent Systems and Applications*, IEEE CS Press. IEEE, IEEE CS Press, October 1999.

[Ken01] E. A. Kendall. Agent software engineering with role modelling. In P. Ciancarini and M. Wooldridge, editors, *Agent-Oriented Software Engineering - Proceedings of the 1st International Workshop AOSE-2000*, number 1957 in LNCS, pages 163–170, Heidelberg, Berlin, 2001. Springer Verlag.

[KJ98] M. Knapik and J. Johnson. *Develping Intelligent Agents for Distributed Systems.* McGraw-Hill, New York, USA, 1998.

[KKO95] T. Ishida K. Kuwabara and N. Osato. Agentalk: Describing multi-agent coordination protocols with inheritance. In *Proceedings of the Seventh IEEE International Conference on Tools with Artificial Intelligence (ICTAI-95)*, pages 460–465, Herndon, Virginia, November 1995. IEEE.

[Klu99] M. Klusch. *Intelligent Information Agents.* Springer Verlag, Heidelberg, Berlin, 1999.

[Koe68] A. Koestler. *The Ghost in the Machine.* The Macmillan Company, 1968.

[Kon99] J. L. Koning. Algorithms for transating interaction protocols into a formal description. In K. Ito, editor, *IEEE International Conference on Systems, Man, and Cybernetics Conference (SMC-99)*, Tokyo, Japan, October 1999. IEEE.

[Kra01] S. Kraus. *Strategic Negotiation in Multiagent Environments.* MIT Press, Cambridge, MA, 2001.

[LF97] Y. Labrou and T. Finin. *Semantics and Conversations for an Agent Communication Language*. Morgan Kaufmann, 1997.

[LF98] Y. Labrou and T. Finin. Semantics for an agent communication language. In M. P. Singh, A. Rao, and M. J. Wooldridge, editors, *Intelligent Agents IV*, volume 1365, pages 209–214. Springer-Verlag: Heidelberg, Germany, 1998.

[LFP99] Y. Labrou, T. Finin, and Y. Peng. Agent communication languages: The current landscape. *IEEE Intelligent Systems*, 14(2):45–52, / 1999.

[LH93] P. B. Luh and D. J. Hoitomt. Scheduling of manufacturing systems using the lagrangian relaxation technique. volume 38 of *IEEE Transactions on Automatic Control*, pages 1066–1080. IEEE, July 1993.

[LMS98] D. Levi, P. Meyer, and B. Stewart. RFC 2273: SNMPv3 applications, January 1998.

[LS99] D. Levi and J. Schönwälder. RFC 2592: Definitions of managed objects for the delegation of management scripts, May 1999.

[Lyn94] N. A. Lynch et al. *Atomic Transactions*. Morgan Kaufmann Series on Data Management Systems. Morgan Kaufmann Publishers, San Mateo, USA, 1994.

[McC96a] K. McCloghrie. RFC 1909: An administrative infrastructure for SNMPv2, February 1996.

[McC96b] K. McCloghrie. RFC 2011: SNMPv2 management information base for the Internet Protocol using SMIv2, November 1996.

[McC96c] K. McCloghrie. RFC 2012: SNMPv2 management information base for the transmission control protocol using SMIv2, November 1996.

[McC96d] K. McCloghrie. RFC 2013: SNMPv2 management information base for the user datagram protocol using SMIv2, November 1996.

[McG03] D. L. McGuinness et al. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2003.

[MES00] B. Moore, E. Ellesson, and J. Strassner. Policy core information model - version 1 specification. Draft, IETF, May 2000.

[MF03] J. P. Martin-Flatin. *Web-Based Management of IP Networks and Systems*. Wiley Series in Communications Networking & Distributed Systems. John Wiley & Sons, LTD, England, 1 edition, January 2003.

[Min75] M. Minsky. *A Framework for Representing Knowledge*. McGraw-Hill, 1975.

[Min85] M. Minsky. *Readings in Knowledge Representation*. Morgan Kaufmann, Los Altos, 1985.

[MLF95] J. Mayfield, Y. Labrou, and T. Finin. Desiderata for agent communication languages. In *AAAI Spring Symposium on Information Gathering*, 1995.

[Mou98] M. A. Mountzia. *A Distributed Management Approach Based on Flexible Agents*, volume 1, pages 99–120. Interoperable Communication Networks, 1998.

[MR90] K. McCloghrie and M. T. Rose. RFC 1156: Management information base for network management of TCP/IP-based internets, May 1990.

[MR91] K. McCloghrie and M. T. Rose. RFC 1213: Management information base for network management of TCP/IP-based internets:MIB-II, March 1991.

[MWD98] J-J. Ch. Meyer, R. J. Wieringa, and F. Dignum. The role of deontic logic in the specification of information systems. In *Logics for Databases and Information Systems*, pages 71–115, 1998.

[NNL98] H. Nwana, D. Ndumu, and L. Lee. Zeus: An advanced tool-kit for engineering distributed multi-agent systems, 1998.

[OJ96] G. M. P. OHare and N. R. Jennings. *Foundations of Distributed Artificial Intelligence*. John Wiley and Sons, 1996.

[OMG01a] OMG, 2001. UML 1.4 Specification. Technical Report, Object Management Group.

[OMG01b] OMG. Common object analysis and design facility (OA&DF) metamodel. Technical report, OMG, 2001.

[OMG01c] OMG. Unified modeling language specification version 1.4. Technical report, OMG, 2001.

[OMG02] OMG. Meta-object facility (MOF) specification. Technical report, Object Management Group, 2002.

[Omi01] A. Omicini. *Societies and Infrastructures in the Analysis and Design of Agent-Based Systems*. International Journal of Software Engineering and Knowledge Engineering. World Scientific Publishing Company, 2001.

[OPB00a] J. Odell, H. Parunak, and B. Bauer. Extending UML for agents, 2000.

[OPB00b] J. Odell, H. V. D. Parunak, and B. Bauer. Representing agent interaction protocols in UML. In *AOSE*, pages 121–140, 2000.

[OPF03] J. Odell, H. Van Dyke Parunak, and M. Fleischer. *Software Engineering for Large-Scale Multi-Agent Systems*. LNCS. Springer Verlag, 2003.

[Pav99] J. Pavón. *Building Telecommunications Management Applications with CORBA*, volume 2 of *IEEE Communications Surveys*, pages 2–16. IEEE, 1999.

[Plo81] G. Plotkin. A structural approach to operational semantics. Technical Report Technical Report DAIMI FN-19, Aarhus University, Computer Science Department, 1981.

[PO01] H. V. D. Parunak and J. Odell. Representing social structures in UML. In *Proceedings of the Autonomous Agents conference*, Montreal, Canada, 2001. ACM.

[Pop98] A. Pope. *The CORBA Reference Guide*. Addison-Wesley, 1998.

[Pör70] I. Pörn. *Action Theory and Social Science*, volume 120 of *Synthese Library*. Dordrecht D. Reidel, 1970.

[PT00] A. Puliafito and O. Tomarchio. *Using Mobile Agents to implement flexible Network Management strategies*, volume 23, pages 708–719. Computer Communication Journal, April 2000.

[PTBH98] J. Pavón, J. Tomas, Y. Bardout, and L. H. Hauw. *CORBA for Network and Service Management in the TINA Framework*, volume 36 of *IEEE Communications Magazine*, pages 72–79. IEEE, March 1998.

[Qui66] R. Quillian. *Semantic Memory.* PhD thesis, Carnegie Mellon University, Pittsburgh (Pensylvania), USA, 1966.

[Qui67] M. Ross Quillian. Word concepts: A theory and simulation of some basic capabilities. *Behavioral Science*, 12:410–430, 1967.

[Raf02] L. Rafalow. CIM policy model white paper. White Paper DSP0108, Distributed Management Task Force, March 2002.

[RBP+91] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen. *Object-Oriented Modeling and Design.* Prentice Hall, 1991.

[RD00] M. G. Rubinstein and O. C. Duarte. *Evaluating Tradeoffs of Mobile Agents in Network Management*, volume 99. Networking and Information Systems, 2000.

[RJB99] J. Rumbaugh, I. Jacobson, and G. Booch. *The Unified Modeling Language Reference Manual.* Object Technology Series. Addison-Wesley, 1999.

[RM90] M. T. Rose and K. McCloghrie. RFC 1155: Structure and identification of management information for TCP/IP-based internets, May 1990.

[RM91] M. T. Rose and K. McCloghrie. RFC 1212: Concise MIB definitions, March 1991.

[Ros96] M. T. Rose. *The Simple Book: An Introduction to Networking Management.* Series in Innovative Technology. Prentice Hall, 2 edition, 1996.

[RV] C. Russ and G. Vierke. Agent-based configuration of virtual enterprises.

[SAL+01] J. Soriano, F. Alonso, G. López, F. Fernández, and P. Rojas. Intelligent virtual agent societies on the internet. *LNCS, Springer-Verlag: Heidelberg, Germany*, (2190), 2001.

[SAL02] J. Soriano, F. Alonso, and G. López. Social commitment policies for formally specifying the organisation and behaviour of open agent societies. In R. Unland, H. Tianfield, and H. Czap, editors, *Proceedings of the 3rd International Symposium on Multi-Agent Systems, Large Complex Systems and E-Business (MALCEB-02)*, 2002.

[SAL03] J. Soriano, F. Alonso, and G. López. A formal specification language for agent conversations. *LNCS, Springer-Verlag: Heidelberg, Germany*, (2691):214–226, 2003.

[SCM+96a] SNMPv2 Working Group, J. Case, K. McCloghrie, M. Rose, and S. Waldbusser. RFC 1901: Introduction to community-based SNMPv2, January 1996.

[SCM+96b] SNMPv2 Working Group, J. Case, K. McCloghrie, M. Rose, and S. Waldbusser. RFC 1902: Structure of management information for version 2 of the Simple Network Management Protocol (SNMPv2), January 1996.

[SCM+96c] SNMPv2 Working Group, J. Case, K. McCloghrie, M. Rose, and S. Waldbusser. RFC 1903: Textual conventions for version 2 of the Simple Network Management Protocol (SNMPv2), January 1996.

[SCM+96d] SNMPv2 Working Group, J. Case, K. McCloghrie, M. Rose, and S. Waldbusser. RFC 1904: Conformance statements for version 2 of the Simple Network Management Protocol (SNMPv2), January 1996.

[SCM+96e] SNMPv2 Working Group, J. Case, K. McCloghrie, M. Rose, and S. Waldbusser. RFC 1905: Protocol operations for version 2 of the Simple Network Management Protocol (SNMPv2), January 1996.

[SCM+96f] SNMPv2 Working Group, J. Case, K. McCloghrie, M. Rose, and S. Waldbusser. RFC 1906: Transport mappings for version 2 of the Simple Network Management Protocol (SNMPv2), January 1996.

[SCM+96g] SNMPv2 Working Group, J. Case, K. McCloghrie, M. Rose, and S. Waldbusser. RFC 1907: Management information base for version 2 of the Simple Network Management Protocol (SNMPv2), January 1996.

[SCM+96h] SNMPv2 Working Group, J. Case, K. McCloghrie, M. Rose, and S. Waldbusser. RFC 1908: Coexistence between version 1 and version 2 of the Internet-standard Network Management Framework, January 1996.

[Sia91] S. S. Sian. *Adaptation based on cooperative learning in multi-agent systems.* Elsevier Science Publishers B.V., 1991.

[Sie96] J. Siegel. *CORBA Fundamentals and Programming.* John Wiley and Sons, 1996.

[Sim90] H. A. Simon. *The Sciences of the Artificial.* MIT Press, Cambridge, MA, 6 edition, 1990.

[Sin97] M. P. Singh. A semantics for speech acts. In M. N. Huhns and M. P. Singh, editors, *Readings in Agents*, pages 458–470. Morgan Kaufmann, San Francisco, CA, USA, 1997.

[SL95] T. W. Sandholm and V. R. Lesser. Advantages of a leveled commitment contracting protocol. Technical Report TR 95-72, University of Massachusetts at Ambherst, Computer Science, 1995.

[SL01] T. W. Sandholm and V. R. Lesser. Leveled commitment contracts and strategic breach. 2001.

[SLA02] J. Soriano, G. López, and F. Alonso. Policy infrastructure as an extension to the FIPA abstract architecture for open agent platform design. In J. P. Müller and B. Bauer, editors, *Proceedings of the International Workshop on Agents and Software Engineering (AGES-02)*, 2002.

[Slo94] M. Sloman, editor. *Network and Distributed Systems Management.* Addison-Wesley, 1994.

[Spi92] M. Spivey. *The Z Notation.* Prentice-Hall International, Hemel Hempstead, UK, 2 edition, 1992.

[SRA02] C. Sierra and J. A. Rodriguez-Aguilar. *Socially Intelligent Agents: Creating Relationships with Computers and Robots*, chapter Enabling Open Agent Institutions, pages 259–266. Kluwer Academic Publishers, 2002.

[Ste94] W. R. Stevens. *TCP/IP Illustrated. Vol. I: The Protocols.* Addison-Wesley, 1994.

[SV85] J. R. Searle and D. Vanderveken. *Foundations of Illocutionary Logic.* Cambridge University Press, Cambridge, England, 1985.

[Tra99] D. R. Traum. *Speech Acts for Dialogue Agents*, pages 169–201. Kluwer Academic Publishers, 1999.

[TSS+97] D. L. Tennenhouse, J. M. Smith, W. D. Sincoskie, D. J. Wetherall, and G. J. Minden. *A Survey of Active Network Research*, volume 35 of *IEEE Communications Magazine*, pages 80–86. IEEE, 1997.

[Tur93] K. J. Turner. *Using Formal Description Techniques. An Introduction to Estelle, LOTOS and SDL.* John Wiley and Sons, Ltd, 1993.

[Udu00] D. K. Udupa. *TMN Telecommunications Management Network.* Telecommunications. McGraw-Hill, New York, NY, 2000.

[UWB01] M. Ulieru, S. S. Walker, and R. W. Brennan. Holonic enterprise as a collaborative information ecosystem. Technical Report SMCB-E-05192001-0063, FIPA Meeting, April 2001. IEEE Transactions of Systems, Man and Cybernetics.

[Ver01] D. C. Verma. *Policy-Based Networking. Architecture and Algorithms.* New Riders Publishing, 2001.

[VPK97] N. Vassila, G. Pavlou, and G. Knight. *Active Objects in TMN.* Integrated Network Management V. Chapman & Hall, 1997.

[Wal95] S. Waldbusser. RFC 1757: Remote network monitoring management information base, February 1995.

[Wal97a] S. Waldbusser. Remote network monitoring management (RMON) information base version 2 using smiv2. Specification RFC 1757, IETF, January 1997.

[Wal97b] S. Waldbusser. RFC 2021: Remote network monitoring management information base version 2 using SMIv2, January 1997.

[Wat86] D. A. Waterman. *A Guide to Expert Systems.* Addison-Wesley, Masachusets, USA, 1986.

[Wat96] G. Waters. RFC 1910: User-based security model for SNMPv2, February 1996.

[WC00] M. Wooldridge and P. Ciancarini. Agent-Oriented Software Engineering: The State of the Art. In P. Ciancarini and M. Wooldridge, editors, *First Int. Workshop on Agent-Oriented Software Engineering*, volume 1957, pages 1–28. Springer-Verlag, Berlin, 2000.

[WC01] M. Wooldridge and P. Ciancarini. Agent-oriented software engineering. In S. K. Chang, editor, *Handbook of Software Engineering and Knowledge Engineering*, page (to appear). World Scientific Publishing Co., 2001.

[Wei99] G. Weiss, editor. *Multi-Agent Systems: A Modern Approach to Distributed Artificial Intelligence*. MIT Press, Cambridge, MA, 1999.

[Wes03] A. Westerinen. What is policy and what can it be?. (keynote). In *Proceedings of the IEEE Policy 2003 Conference*. IEEE Computer Society Press, 2003.

[WF86] T. Winograd and F. Flores. *Understanding Computers and Cognition*. Addison-Wesley, 1986.

[WG01] C. A. Welty and N. Guarino. Supporting ontological analysis of taxonomic relationships. *Data Knowledge Engineering*, 39(1):51–74, 2001.

[WG03a] W3C RDF Core WG. RDF semantics. Working draft, World Wide Web Consortium, January 2003.

[WG03b] W3C WebOnt WG. Web ontology language (owl) guide. Last call working draft, World Wide Web Consortium, 2003.

[Wie95] R. Wies. *Policies in Integrated Network and Systems Management: Methodologies for the Definition, Transformation and Application of Management Policies*. PhD thesis, University of Munich, 1995.

[WJK00] M. Wooldridge, N. R. Jennings, and D. Kinny. The gaia methodology for agent-oriented analysis and design. *Autonomous Agents and Multi-Agent Systems*, 3(3):285–312, 2000.

[WM89] R. J. Wieringa and J-J. Ch. Meyer. *Applications of Deontic Logic in Computer Science: A Concise Overview*, pages 17–40. 1989.

[Woo99] M. Wooldridge. Intelligent agents. In Gerhard Weiss, editor, *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, pages 27–78. The MIT Press, Cambridge, MA, USA, 1999.

[Woo00] M. Wooldridge. *Reasoning about Rational Agents*. MIT Press, Cambridge, MA, 2000.

[WPB99] T. White, B. Pagurek, and A. Bieszczad. *Network Modeling for Management Applications Using Intelligent Mobile Agents*. Journal of Network and Systems Management, 1999.

[WPM98] B. Wijnen, R. Presuhn, and K. McCloghrie. RFC 2275: View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP), January 1998.

[WR99] M. Wooldridge and A. Rao, editors. *Foundations and Theories Of Rational Agents*. Kluwer Academic Publishers, 1999.

[WWC01] M. Wooldridge, G. Weiss, and P. Ciancarini, editors. *Proceedings of the 2nd International Workshop (AOSE2001)*, volume 2222 of *LNCS*, Heidelberg, Berlin, 2001. Springer Verlag.

[WWMD91] R. J. Wieringa, H. Weigand, J-J. Ch. Meyer, and F. Dignum. The inheritance of dynamic and deontic integrity constraints. *Annals of Mathematics and Artificial Intelligence*, 3(2–4):393–428, 1991.

[Yem93] Y. Yemini. The OSI network management model. *IEEE Communication Magazine*, (31):20–29, May 1993.

[ZCPZ97] T. Zhang, S. Covaci, and R. Popescu-Zeletin. Intelligent agents in network and service management. In *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM'97)*, volume 1, pages 202–206, Phoenix, AZ, USA, November 1997. IEEE, IEEE.

[ZJOW00] F. Zambonelli, N. R. Jennings, A. Omicini, and M. Wooldridge. Agent-Oriented Software Engineering for Internet Applications. In A. Omicini, F. Zambonelli, M. Klusch, and R. Tolksdorf, editors, *Coordination of Internet Agents: Models, Technologies, and Applications*, pages 326–346. Springer-Verlag: Heidelberg, Germany, 2000.

[ZJW01] F. Zambonelli, N. R. Jennings, and M. Wooldridge. Organisational rules as an abstraction for the analysis and design of multi-agent systems. *International Journal of Software Engineering and Knowledge Engineering*, 11(3):303–328, 2001.

Apéndice A

"Mapping" de CIM a OWL

Índice General

A.1. Complejidad del modelo CIM 209

A.2. Expresión en $\mathcal{ALE}\mathcal{CNOQ}^{-}_{\mathcal{HR}^{+}\circ}$ del Modelo Central de CIM 210

A.3. Especificación OWL del Modelo de Políticas CIM 220

A.4. Especificación $\mathcal{ALE}\mathcal{CNOQ}^{-}_{\mathcal{HR}^{+}\circ}$ del Modelo de Políticas CIM 234

Este apéndice recoge el "mapping" de un conjunto de modelos CIM a OWL. En primer lugar se presenta un cuadro con los principales componentes de cada uno de los modelos CIM desarrollados por el DMTF. Dicho cuadro muestra la complejidad de estos modelos en términos de número de clases, número de asociaciones y número de atributos. A continuación se especifica el "mapping" del *Modelo Central* de CIM expresado con la sintaxis abstracta descrita en el capítulo 3. Finalmente se presenta el "mapping" del *Modelo de Políticas* de CIM expresado tanto con la sintaxis abstracta mencionada anteriormente (y por tanto en lógica descriptiva), como en OWL-FULL. Este último ejemplo permite evaluar las diferencias existentes entre ambos tipos de "mapping" (CIM/OWL-DL y CIM/OWL-FULL) en términos de expresividad.

A.1. Complejidad del modelo CIM

El cuadro A.1 recoge los principales componentes de cada uno de los modelos CIM desarrollados por el DMTF en su versión 2.6. El cuadro muestra la complejidad asociada a la formalización de estos modelos en términos de número de clases, número de asociaciones (dependencias, asociaciones y agregaciones), y número de

atributos. Siguiendo la filosofía del meta-esquema CIM, las asociaciones se contemplan como un tipo de clase.

A.2. Expresión en $\mathcal{ALE}\mathcal{CNOQ}^{-}_{\mathcal{HR}^{+}\circ}$ del Modelo Central de CIM

A continuación se describe la expresión en lógica descriptiva $\mathcal{ALE}\mathcal{CNOQ}^{-}_{\mathcal{HR}^{+}\circ}$ del *Modelo Central* de CIM recogido en las figuras A.1 y A.2.

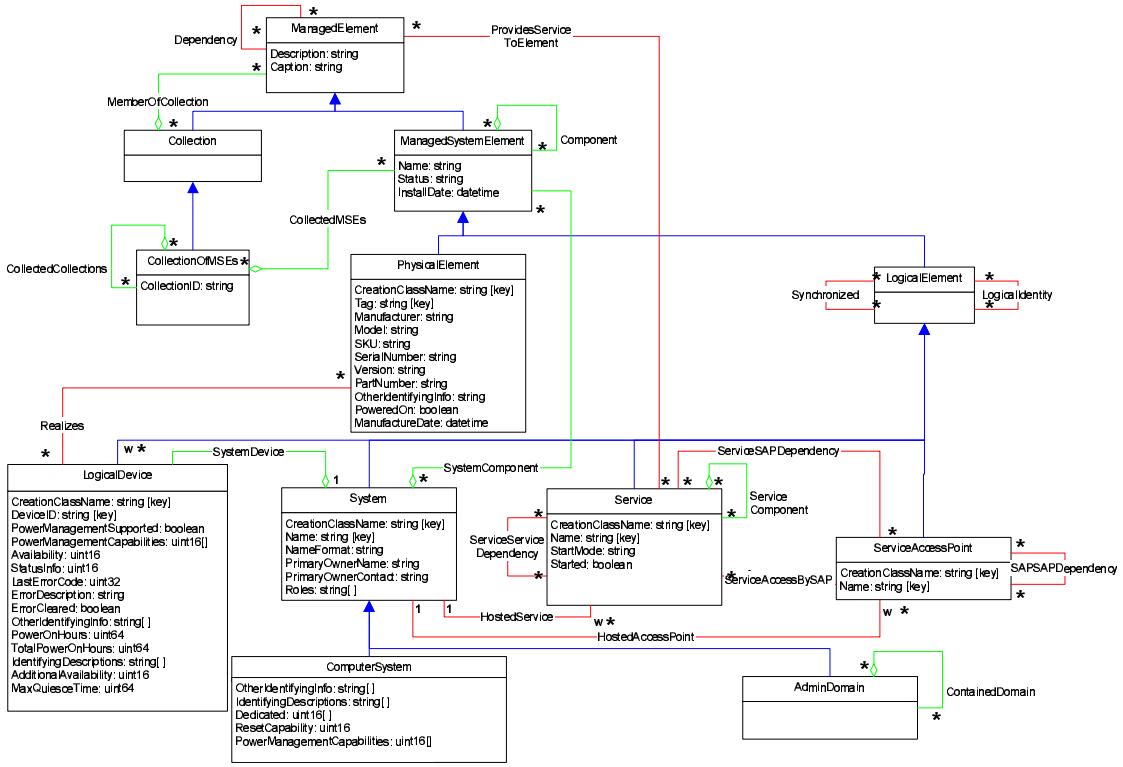

Figura A.1: Modelo Central de CIM versión 2.6 (DMTF SysDev-WG)

$$
\begin{aligned}
ManagedElement \sqsubseteq & Class \sqcap \exists Description.string \sqcap \langle \leq 1 Description \rangle \sqcap \\
& \exists Caption.string \sqcap \langle \leq 1 Caption \rangle \sqcap \\
& \forall Dependency\_Antecedent^{-}.Dependency \sqcap \\
& \forall Dependency\_Dependent^{-}.Dependency \sqcap \\
& \forall DependencyRole.Dependency \sqcap \\
& \forall ProvidesServiceToElement\_Dependent^{-}.ProvidesServiceToElement \sqcap \\
& \forall ProvidesServiceToElementRole^{-}.Service \sqcap \\
& \forall MemberOfCollection\_Part^{-}.MemberOfCollection \sqcap \\
& \forall MemberOfCollectionRole^{-}.Collection
\end{aligned}
$$

Especificación CIM	**Clases**	**Atrib.**
CIM Core Specification. DMTF SysDev-WG		
Elementos lógicos y sus subclases	12	37
Elementos físicos y dispositivos lógicos	4	39
Colecciones y conjuntos de redundancia	3	2
Grupos de redundancia	4	9
Producto, FRU e identidad software	4	28
Estadísticas	7	13
Información de ajuste, perfiles, capacidades y gestión de energía	7	7
Ajustes, configuraciones y parámetros	5	7
Jerarquías de asociación, dependencia y agregación	78	21
CIM User Specification. DMTF User-WG		
Credenciales y autenticación	5	16
Servicios de seguridad	7	1
Servicios de seguridad Kerberos	2	4
Servicios de seguridad de clave pública	4	12
control de acceso	3	14
Organización y persona	8	65
Grupos y roles	4	41
Cuentas	2	12
Jerarquías de dependencia y agregación	45	11
CIM Database Specification. DMTF Database-WG		
Entorno de base de datos	6	13
Software y estadísticas	3	21
Jerarquía de asociación	5	3
CIM Device Specification. DMTF SysDev-WG		
Disipación y energia	10	58
Procesadores	3	19
Controladores y controladores PCI	18	88
Puertos lógicos y grupos de puertos	12	62
Adaptadores de red	7	12
Canales de fibra	7	35
Dispositivos de almacenamiento y extensiones	18	69
Modelo extendido SCC	8	10
Servicios y bibliotecas de almacenamiento	18	118
Dispositivos de interacción con el usuario	8	30
Memoria	4	28
Modems	12	80

Cuadro A.1: Complejidad de la especificación CIM en términos de número de clases, atributos y relaciones

Especificación CIM	**Clases**	**Atrib.**
CIM Device Specification. DMTF SysDev-WG (Cont.)		
Impresoras, trabajos y servicios de impresión	6	67
Sensores y alarmas	10	43
Grupo de USB y disco	5	16
Jerarquías de dependencias, asociaciones y agregaciones	89	82
CIM Event Specification. DMTF Event/Interop-WG		
Clases indicación	16	39
Filtros y manejadores de indicaciones	3	15
Jerarquía de asociaciones	1	16
CIM Interop Schema. DMTF Interop-WG		
Esquema de interoperabilidad	8	36
Jerarquía de asociaciones	9	2
CIM Metrics Schema. DMTF Application-WG		
Clases de métricas y unidades de trabajo (UoW)	7	31
Jerarquía de asociaciones	12	0
CIM Network Specification. DMTF Networks-WG		
Sistemas y colecciones en red	14	33
Terminaciones de protocolo	12	45
Encaminamiento y reenvío	6	19
Rutas y *Pipes*	6	37
Filtrado y entradas de filtrado	6	36
Memorias intermedias (recursos de red)	1	7
Gestión SNMP	3	7
Encaminamiento OSPF	5	21
Encaminamiento BGP	9	54
Puentes y *switches*	10	47
Redes virtuales (VLAN)	3	1
Calidad de servicio (QoS)	32	71
Jerarquías de dependencias, asociación y agregación	115	46
CIM Physical Specification. DMTF SysDev-WG		
Paquetes físicos, paquetes de almacenamiento, conectores y enlaces	13	73
Componente físico	5	27
Capacidad física y conjuntos de reemplazo	4	12
Estadísticas de almacenamiento	2	18
Jerarquías de asociación y agregación	23	7

Cuadro A.2: Complejidad de la especificación CIM en términos de número de clases, atributos y relaciones (Cont.)

CIM Policy Specification. DMTF Policy-WG		
Conjuntos de políticas	4	17
Condiciones	4	15
Acciones	3	10
Clases de sistema	1	0
Colección de roles de políticas	3	2
Jerarquías de dependencias y agregaciones	19	5
CIM Support Specification		
Contacto	10	40
Elemento de expresión	7	19
Solución	5	7
Incidente de servicio	5	26
Jerarquías de dependencias y agregaciones	35	0
CIM System Specification. DMTF SysDev-WG		
Ordenador	9	26
Sistemas de ficheros	10	42
Software	0	0
Sistema operativo	1	23
Procesamiento y trabajos	5	33
Servicio de inicialización	4	17
Tiempo	1	16
UNIX	7	59
Recursos del sistema	7	23
Mensajes de log	2	25
Diagnósticos	3	38
Jerarquías de asociaciones, dependencias y agregaciones	61	13
CIM Application Model Specification. DMTF Application-WG		
Características del software instalado	5	24
Comprobación y acciones del software	21	54
Manejadores de dispositivos y software de BIOS	5	15
Jerarquía de asociaciones	30	2
Totales	**1.069**	**2.444**

Cuadro A.3: Complejidad de la especificación CIM en términos de número de clases, atributos y relaciones (Cont.)

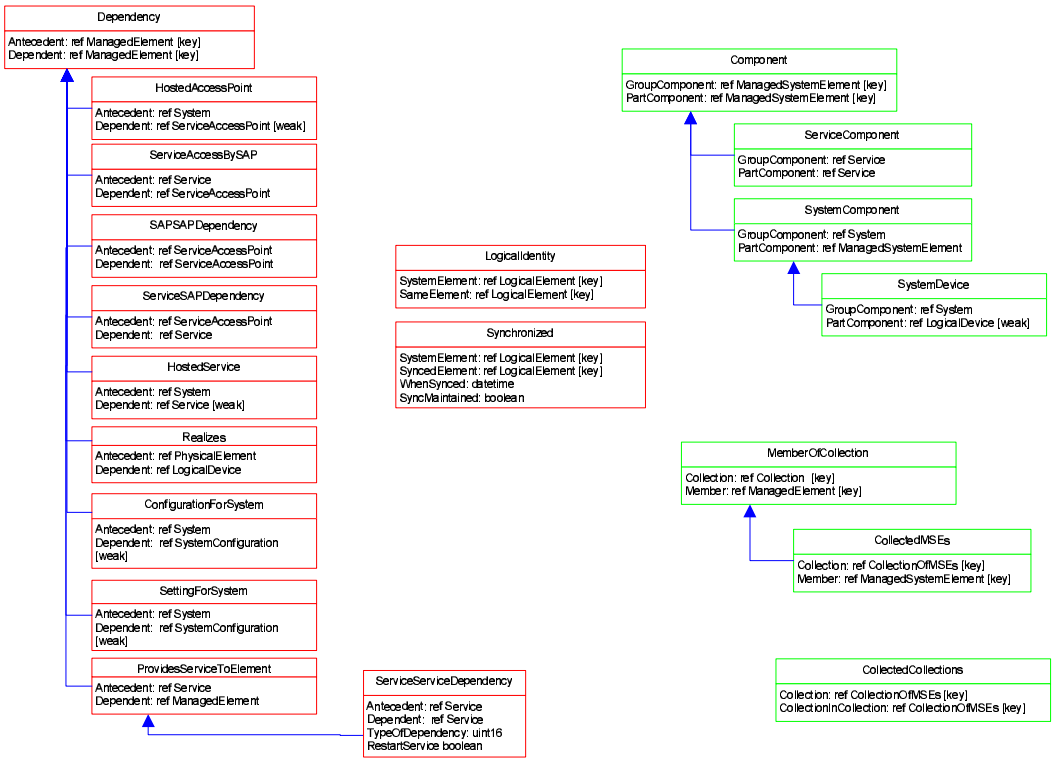

Figura A.2: Jerarquía de herencia de las clases asociación presentes en el *Modelo Central* de CIM como instancias de la meta-clase ASSOCIATION (DMTF SysDev-WG)

$$Dependency \sqsubseteq Association \sqcap \exists Dependency\_Antecedent.ManagedElement \sqcap \langle \leq 1 Dependency\_Antecedent \rangle \exists Dependency\_Dependent.ManagedElement \sqcap \langle \leq 1 Dependency\_Dependent \rangle$$

$$DependencyRole \equiv Dependency\_Antecedent^{-} \circ Dependency\_Dependent$$

$$ProvidesServiceToElement \sqsubseteq Association \sqcap \exists ProvidesServiceToElement\_Antecedent.Service \sqcap \langle \leq 1 ProvidesServiceToElement\_Antecedent \rangle \exists ProvidesServiceToElement\_Dependent.ManagedElement \sqcap \langle \leq 1 ProvidesServiceToElement\_Dependent \rangle$$

$$ProvidesServiceToElementRole \equiv ProvidesServiceToElement\_Antecedent^{-} \circ ProvidesServiceToElement\_Dependent$$

$$MemberOfCollection \sqsubseteq Aggregation \sqcap \exists MemberOfCollection\_Group.Collection \sqcap \langle \leq 1 MemberOfCollection\_Group \rangle \sqcap \langle \leq 1 MemberOfCollection\_Part \rangle \sqcap \exists MemberOfCollection\_Part.ManagedElement$$

$$MemberOfCollectionRole \equiv MemberOfCollection\_Group^{-} \circ MemberOfCollection\_Part$$

$$ManagedSystemElement \sqsubseteq Class \sqcap ManagedElement \sqcap \exists Name.string \sqcap \langle \leq 1 Name \rangle \sqcap \exists Status.string \sqcap \langle \leq 1 Status \rangle \sqcap \exists InstallDate.datetime \sqcap \langle \leq 1 InstallDate \rangle \sqcap \forall Component\_Group^{-}.ManagedSystemElement \sqcap \forall Component\_Part^{-}.ManagedSystemElement \sqcap \forall ComponentRole.ManagedSystemElement \sqcap \forall CollectedMSEs\_Part^{-}.CollectedMSEs \sqcap \forall CollectedMSEsRole^{-}.CollectionOfMSEs \sqcap \forall SystemComponent\_Part^{-}.SystemComponent \sqcap \forall SystemComponentRole^{-}.System$$

$$ComponentRole \equiv Component\_Group^{-} \circ Component\_Part$$

$$Component \sqsubseteq Aggregation \sqcap \exists Component\_Group.ManagedSystemElement \sqcap \exists Component\_Part.ManagedSystemElement \sqcap \langle \leq 1 Component\_Group \rangle \sqcap \langle \leq 1 Component\_Part \rangle$$

$$Collection \sqsubseteq Class \sqcap ManagedElement \sqcap \forall MemberOfCollectionRole.ManagedElement \sqcap \forall MemberOfCollection\_Group^{-}.MemberOfCollection$$

$$CollectionOfMSEs \sqsubseteq Class \sqcap Collection \sqcap \exists CollectionID.string \sqcap \langle \leq 1 CollectionID \rangle \sqcap \forall s\_Group^{-}.CollectionOfMSEs \sqcap \forall s\_Part^{-}.CollectionOfMSEs \sqcap \forall CollectedMSEs\_Group^{-}.CollectedMSEs \sqcap \forall CollectedMSEsRole.ManagedSystemElement$$

$$CollectedMSEs \sqsubseteq MemberOfCollection \sqcap \exists CollectedMSEs\_Group.CollectionOfMSEs \sqcap \langle \leq 1 CollectedMSEs\_Group \rangle \sqcap \langle \leq 1 CollectedMSEs\_Part \rangle \sqcap \exists CollectedMSEs\_Part.ManagedSystemElement$$

$$CollectedMSEsRole \equiv CollectedMSEs\_Group^{-} \circ CollectedMSEs\_Part$$

$$
\begin{aligned}
PhysicalElement \sqsubseteq & Class \sqcap ManagedSystemElement \sqcap \\
& \exists CreationClassName.string \sqcap \\
& \langle \leq 1 CreationClassName \rangle \sqcap \exists Tag.string \sqcap \langle \leq 1 Tag \rangle \sqcap \\
& \exists Manufacturer.string \sqcap \langle \leq 1 Manufacturer \rangle \sqcap \\
& \exists Model.string \sqcap \langle \leq 1 Model \rangle \sqcap \exists SKU.string \sqcap \langle \leq 1 SKU \rangle \sqcap \\
& \exists SerialNumber.string \sqcap \langle \leq 1 SerialNumber \rangle \sqcap \\
& \exists Version.string \sqcap \langle \leq 1 Version \rangle \sqcap \\
& \exists PartNumber.string \sqcap \langle \leq 1 PartNumber \rangle \sqcap \\
& \exists OtherIdentifyingInfo.string \sqcap \langle \leq 1 OtherIdentifiyingInfo \rangle \sqcap \\
& \exists PoweredOn.boolean \sqcap \langle \leq 1 PoweredOn \rangle \sqcap \\
& \exists ManufactureDate.datetime \sqcap \langle \leq 1 ManufactureDate \rangle \sqcap \\
& \forall \_Group^{-}. \sqcap \forall Role.
\end{aligned}
$$

$$
\begin{aligned}
LogicalElement \sqsubseteq & Class \sqcap ManagedSystemElement \sqcap \\
& \forall Synchronized\_Antecedent^{-}.Synchronized \sqcap \\
& \forall Synchronized\_Dependent^{-}.Synchronized \sqcap \\
& \forall SynchronizedRole.LogicalElement \sqcap \\
& \forall LogicalIdentity\_Antecedent^{-}.LogicalIdentity \sqcap \\
& \forall LogicalIdentity\_Dependent^{-}.LogicalIdentity \sqcap \\
& \forall LogicalIdentityRole.LogicalElement
\end{aligned}
$$

$$
\begin{aligned}
Synchronized \sqsubseteq & Association \sqcap \exists WhenSynced.datetime \sqcap \langle \leq 1 WhenSynced \rangle \sqcap \\
& \exists SyncMaintained.Boolean \sqcap \langle \leq 1 SyncMaintained \rangle \sqcap \\
& \exists Synchronized\_Antecedent.LogicalElement \sqcap \\
& \langle \leq 1 Synchronized\_Antecedent \rangle \sqcap \\
& \exists Synchronized\_Dependent.LogicalElement \sqcap \\
& \langle \leq 1 Synchronized\_Dependent \rangle
\end{aligned}
$$

$$
SynchronizedRole \equiv Synchronized\_Antecedent^{-} \circ Synchronized\_Dependent
$$

$$
\begin{aligned}
LogicalIdentity \sqsubseteq & Association \sqcap \exists LogicalIdentity\_Antecedent.LogicalElement \sqcap \\
& \exists LogicalIdentity\_Dependent.LogicalElement \sqcap \\
& \langle \leq 1 LogicalIdentity\_Antecedent \rangle \sqcap \langle \leq 1 LogicalIdentity\_Dependent \rangle
\end{aligned}
$$

$$
LogicalIdentityRole \equiv LogicalIdentity\_Antecedent^{-} \circ LogicalIdentity\_Dependent
$$

$$
\begin{aligned}
ServiceAccessPoint \sqsubseteq\ & Class \sqcap LogicalElement \sqcap \exists CreationClassName.string \sqcap \\
& \langle \leq 1 CreationClassName \rangle \sqcap \exists Name.string \sqcap \langle \leq 1 Name \rangle \sqcap \\
& \forall SAPSAPDependency\_Antecedent^{-}.SAPSAPDependency \sqcap \\
& \forall SAPSAPDependency\_Dependent^{-}.SAPSAPDependency \sqcap \\
& \forall SAPSAPDependencyRole.ServiceAccessPoint \sqcap \\
& \forall ServiceAccessBySAP\_Dependent^{-}.ServiceAccessBySAP \sqcap \\
& \forall ServiceSAPDependencyRole^{-}.ServiceAccessPoint \sqcap \\
& \forall ServiceSAPDependency\_Dependent^{-}.ServiceSAPDependency \sqcap \\
& \exists HostedAccessPoint\_Dependent^{-}.HostedAccessPoint \sqcap \\
& \langle \leq 1 HostedAccessPoint\_Dependent^{-} \sqcap \\
& \exists HostedAccessPointRole^{-}.System \sqcap \langle \leq 1 HostedAccessPointRole^{-} \rangle
\end{aligned}
$$

$$
\begin{aligned}
Service \sqsubseteq\ & Class \sqcap LogicalElement \sqcap \\
& \exists CreationClassName.string \sqcap \langle \leq 1 CreationClassName \rangle \sqcap \\
& \exists Name.string \sqcap \langle \leq 1 Name \rangle \sqcap \exists StartMode.string \sqcap \\
& \langle \leq 1 StartMode \rangle \sqcap \exists Started.boolean \sqcap \langle \leq 1 Started \rangle \sqcap \\
& \forall ServiceComponent\_Group^{-}.ServiceComponent \sqcap \\
& \forall ServiceComponent\_Part^{-}.ServiceComponent \sqcap \\
& \forall ServiceComponentRole.Service \sqcap \\
& \forall ServiceAccessBySAP\_Antecedent^{-}.ServiceAccessBySAP \sqcap \\
& \forall ServiceSAPDependency\_Antecedent^{-}.ServiceSAPDependency \sqcap \\
& \forall ServiceSAPDependencyRole.ServiceAccessPoint \sqcap \\
& \exists HostedService\_Dependent^{-}.HostedService \sqcap \langle \leq 1 HostedService\_Dependent^{-} \rangle \sqcap \\
& \exists HostedServiceRole^{-}.System \sqcap \langle \leq 1 HostedServiceRole^{-} \rangle \sqcap \\
& \forall ProvidesServiceToElement\_Antecedent^{-}.ProvidesServiceToElement \sqcap \\
& \forall ProvidesServiceToElementRole.ManagedElement \sqcap \\
& \forall ServiceServiceDependency\_Antecedent^{-}.ServiceServiceDependency \sqcap \\
& \forall ServiceServiceDependency\_Dependent^{-}.ServiceServiceDependency \sqcap \\
& \forall ServiceServiceDependencyRole.Service
\end{aligned}
$$

$$
\begin{aligned}
ServiceAccessBySAPRole \equiv\ & ServiceAccessBySAP\_Antecedent^{-} \circ \\
& ServiceAccessBySAP\_Dependent
\end{aligned}
$$

$$
\begin{aligned}
ServiceAccessBySAP \sqsubseteq\ & Association \sqcap \exists ServiceAccessBySAP\_Antecedent.Service \sqcap \\
& \exists ServiceAccessBySAP\_Dependent.ServiceAccessPoint \sqcap \\
& \langle \leq 1 ServiceAccessBySAP\_Antecedent \rangle \sqcap \\
& \langle \leq 1 ServiceAccessBySAP\_Dependent \rangle
\end{aligned}
$$

$$
\begin{aligned}
ServiceSAPDependencyRole \equiv\ & ServiceSAPDependency\_Antecedent^{-} \circ \\
& ServiceSAPDependency\_Dependent
\end{aligned}
$$

$$ServiceSAPDependency \sqsubseteq Association \sqcap \exists ServiceSAPDependency\_Antecedent.Service \sqcap \langle \leq 1 ServiceSAPDependency\_Antecedent \rangle \exists ServiceSAPDependency\_Dependent.ServiceAccessPoint \sqcap \langle \leq 1 ServiceSAPDependency\_Dependent \rangle$$

$$ServiceComponentRole \equiv ServiceComponent\_Group^{-} \circ ServiceComponent\_Part$$

$$SAPSAPDependency \sqsubseteq Association \sqcap \langle \leq 1 SAPSAPDependency\_Antecedent \rangle \sqcap \exists SAPSAPDependency\_Antecedent.ServiceAccessPoint \sqcap \exists SAPSAPDependency\_Dependent.ServiceAccessPoint \sqcap \langle \leq 1 SAPSAPDependency\_Dependent \rangle$$

$$SAPSAPDependency \equiv SAPSAPDependency\_Antecedent^{-} \circ SAPSAPDependency\_Dependent$$

$$ServiceComponent \sqsubseteq Aggregation \sqcap \exists ServiceComponent\_Group.Service \sqcap \langle \leq 1 ServiceComponent\_Group \rangle \sqcap \exists ServiceComponent\_Part.Service \sqcap \langle \leq 1 ServiceComponent\_Part \rangle$$

$$ServiceServiceDependencyRole \equiv ServiceServiceDependency\_Antecedent^{-} \circ ServiceServiceDependency\_Dependent$$

$$ServiceServiceDependency \sqsubseteq Association \sqcap \exists ServiceServiceDependency\_Antecedent.Service \sqcap \langle \leq 1 ServiceServiceDependency\_Antecedent \rangle \sqcap \exists ServiceServiceDependency\_Dependent.Service \sqcap \langle \leq 1 ServiceServiceDependency\_Dependent \rangle$$

$$System \sqsubseteq Class \sqcap LogicalElement \sqcap \exists CreationClassName.string \sqcap \langle \leq 1 CreationClassName \rangle \sqcap \exists Name.string \sqcap \langle \leq 1 Name \rangle \sqcap \exists NameFormat.string \sqcap \langle \leq 1 NameFormat \rangle \sqcap \exists PrimaryOwnerName.string \sqcap \langle \leq 1 PrimaryOwnerName \rangle \sqcap \exists PrimaryOwnerContact.string \sqcap \langle \leq 1 PrimaryOwnerContact \rangle \sqcap \forall Roles.string \sqcap \langle \geq 1 Roles \rangle \sqcap \forall HostedService\_Antecedent^{-}.HostedService \sqcap \forall HostedServiceRole.Service \forall SystemComponent\_Group^{-}.SystemComponent \sqcap \forall SystemComponentRole.ManagedSystemElement \sqcap \forall HostedAccessPoint\_Antecedent^{-}.HostedAccessPoint \sqcap \forall HostedAccessPointRole.ServiceAccessPoint \sqcap \exists SystemDevice\_Group^{-}.SystemDevice \sqcap \langle \leq 1 SystemDevice\_Group^{-} \rangle \sqcap \forall SystemDeviceRole.LogicalDevice$$

$$HostedServiceRole \equiv HostedService\_Antecedent^{-} \circ HostedService\_Dependent$$

$$\begin{aligned} HostedService \sqsubseteq & Aggregation \sqcap \\ & \exists HostedService\_Antecedent.System \sqcap \langle \leq 1 HostedService\_Antecedent \rangle \sqcap \\ & \exists HostedService\_Dependent.Service \sqcap \langle \leq 1 HostedService\_Dependent \rangle \end{aligned}$$

$$\begin{aligned} HostedAccessPointRole \equiv & HostedAccessPoint\_Antecedent^{-} \circ \\ & HostedAccessPoint\_Dependent \end{aligned}$$

$$\begin{aligned} HostedAccessPoint \sqsubseteq & Aggregation \sqcap \\ & \exists HostedAccessPoint\_Antecedent.System \sqcap \\ & \exists HostedAccessPoint\_Dependent.ServiceAccessPoint \sqcap \\ & \langle \leq 1 HostedAccessPoint\_Antecedent \rangle \sqcap \\ & \langle \leq 1 HostedAccessPoint\_Dependent \rangle \end{aligned}$$

$$SystemComponentRole \equiv SystemComponent\_Group^{-} \circ SystemComponent\_Part$$

$$\begin{aligned} SystemComponent \sqsubseteq & Component \sqcap \exists SystemComponent\_Group.System \sqcap \\ & \exists SystemComponent\_Part.ManagedSystemElement \sqcap \\ & \langle \leq 1 SystemComponent\_Group \rangle \sqcap \langle \leq 1 SystemComponent\_Part \rangle \end{aligned}$$

$$\begin{aligned} AdminDomain \sqsubseteq & Class \sqcap System \sqcap \forall ContainedDomainRole.AdminDomain \sqcap \\ & \forall ContainedDomain\_Group^{-}.ContainedDomain \sqcap \\ & \forall ContainedDomain\_Part^{-}.ContainedDomain \end{aligned}$$

$$\begin{aligned} ContainedDomain \sqsubseteq & Aggregation \sqcap \\ & \exists ContainedDomain\_Group.AdminDomain \sqcap \\ & \langle \leq 1 ContainedDomain\_Group \rangle \sqcap \\ & \exists ContainedDomain\_Part.AdminDomain \sqcap \\ & \langle \leq 1 ContainedDomain\_Part \rangle \end{aligned}$$

$$ContainedDomainRole \equiv ContainedDomain\_Group^{-} \circ ContainedDomain\_Part$$

$$\begin{aligned} ComputerSystem \sqsubseteq & Class \sqcap System \sqcap \\ & \forall OtherIdentifyingInfo.string \sqcap \langle \geq 1 OtherIdentifyingInfo \rangle \sqcap \\ & \forall IdentifyingDescriptions.string \sqcap \langle \geq 1 IdentifyingDescriptions \rangle \sqcap \\ & \forall Dedicated.uint16 \sqcap \langle \geq 1 Dedicated \rangle \sqcap \\ & \exists ResetCapability.uint16 \sqcap \langle \leq 1 ResetCapability \rangle \sqcap \\ & \forall PowerManagementCapabilities.uint16 \sqcap \langle \geq 1 PrimaryOwnerContact \rangle \end{aligned}$$

A.3. Especificación OWL del Modelo de Políticas CIM

A continuación se describe la expresión en OWL-FULL del *Modelo de Políticas* de CIM recogido en las figuras A.3 y A.4.

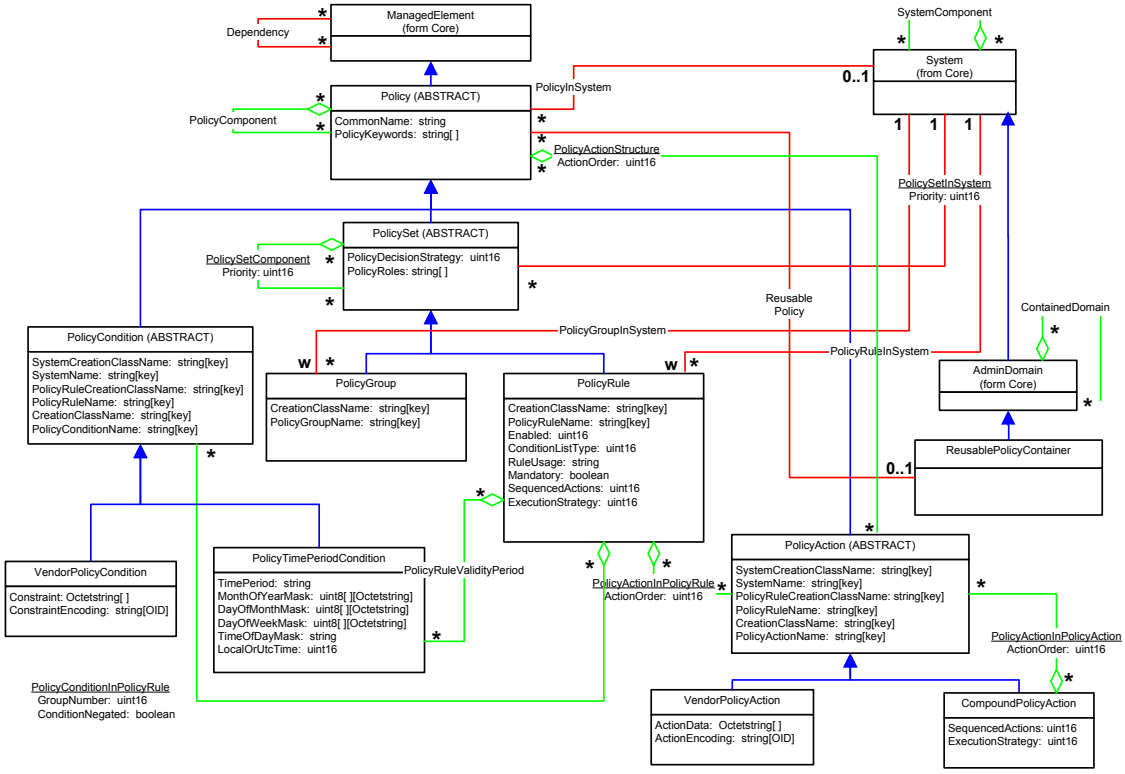

Figura A.3: Modelo de políticas CIM versión 2.6 (DMTF Service Level Agreements WG)

```
<?xml version="1.0" encoding="UTF-8"?> <!DOCTYPE owl [
   <!ENTITY cpm "urn:TIC2001-3451:ontologies:cim-policy-model#">
   <!ENTITY owl "http://www.w3.org/2002/07/owl#">
   <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">
   <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
   <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#">]>
   <rdf:RDF xmlns:rdf="&rdf;" xmlns:rdfs="&rdfs;"
            xmlns:owl="&owl;" xmlns:xsd="&xsd;" xmlns="&acsl;"
            xmlns:acl="&acl;" xmlns:dc="http://purl.org/dc/elements/1.1/">
   <owl:Ontology rdf:about='urn:TIC2001-3451:ontologies:cim-policy-model#'>
      <rdfs:comment>Ontologia para el modelo de politicas de CIM</rdfs:comment>
      <owl:versionInfo>
            $Id: CIMPolicyModelOntology v 1.0 02/02/2003 Javier Soriano $
        </owl:versionInfo>
      <owl:imports rdf:resource='http://www.w3.org/2002/07/owl'/>
      <owl:imports rdf:resource='http://www.w3.org/2000/01/rdf-schema'/>
      <dc:title>OWL CIM Policy Model Ontology</dc:title>
      <dc:creator>Francisco Javier Soriano Camino</dc:creator>
      <dc:subject>OWL; Ontologies; Policy Models</dc:subject>
      <dc:description>Ontologia OWL para el modelo de politicas de CIM 2.6</dc:description>
```

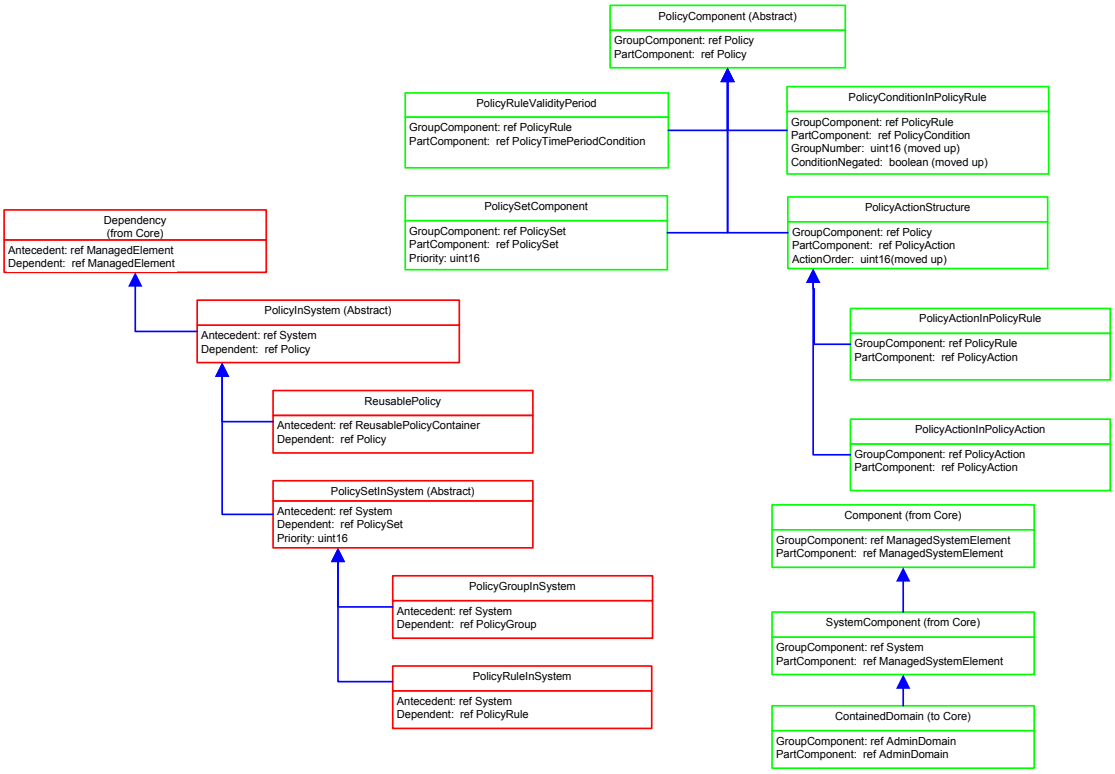

Figura A.4: Jerarquía de herencia de las clases asociación presentes en el modelo de políticas CIM como instancias de la meta-clase ASSOCIATION (DMTF Service Level Agreements WG)

```
    <dc:publisher>Grupo de Redes y Sistemas Distribuidos. Facultad de Informatica</dc:publisher>
    <dc:date>Feb 02, 2003</dc:date>
    <dc:format>text/xml</dc:format>
    <dc:language>es</dc:language>
    <dc:identifier>urn:TIC2001-3451:ontologies:acsl</dc:identifier>
</owl:Ontology>
<owl:Class rdf:ID='KeyProperty'>
    <owl:sameClassAs rdf:resource='owl:InverseFunctionalProperty'/>
</owl:Class>
<owl:DatatypeProperty rdf:ID='isKey'>
    <rdfs:domain rdf:resource='owl:DatatypeProperty'/>
    <rdfs:range rdf:resource='xsd:boolean'/>
</owl:DatatypeProperty>
<owl:Class rdf:ID='DependencyProperty'>
    <owl:functionalSubClassOf rdf:resource='owl:ObjectProperty'/>
</owl:Class>
<owl:Class rdf:ID='AggregationProperty'>
    <rdfs:subClassOf rdf:resource='owl:ObjectProperty'/>
    <owl:disjointWith rdf:resource='DependencyProperty'/>
</owl:Class>
<owl:Class rdf:ID='WeakProperty'>
    <rdfs:subClassOf rdf:resource='owl:ObjectProperty'/>
    <owl:disjointWith rdf:resource='AggregationProperty'/>
</owl:Class>
<owl:DatatypeProperty rdf:ID='CreationClassName'>
    <rdfs:domain rdf:resource='#PolicyRule'/>
    <rdfs:range rdf:resource='xsd:string'/>
</owl:DatatypeProperty>
<!-- ************************************ -->
```

```
<!-- ************************************* -->
<!--                Clase Policy                    -->
<!-- ************************************* -->
<owl:Class rdf:ID='Policy'>
   <rdfs:comment>Clase Policy</rdfs:comment>
   <rdfs:label xml:lang='en'>policy</rdfs:label>
   <rdfs:label xml:lang='es'>pol&acute;itica</rdfs:label>
   <owl:functionalSubClassOf rdf:resource='ccm:ManagedElement'/>
   <rdfs:comment>Propiedad CommonName</rdfs:comment>
   <owl:Restriction>
      <owl:onProperty rdf:resource='#CommonName'/>
      <owl:cardinality>1</owl:cardinality>
   </owl:Restriction>
   <rdfs:comment>Propiedad PolicyKeywords. Lista</rdfs:comment>
   <owl:Restriction>
      <owl:onProperty rdf:resource='#PolicyKeyword'/>
      <owl:minCardinality>1</owl:minCardinality>
   </owl:Restriction>
   <rdfs:comment>Asociacion PolicyComponent. 0..*</rdfs:comment>
   <owl:Restriction>
      <owl:onProperty rdf:resource='#PolicyComponent'/>
      <owl:allValuesFrom rdf:resource='#Policy'/>
   </owl:Restriction>
   <rdfs:comment>Asociacion PolicyComponent. 0..*</rdfs:comment>
   <owl:Restriction>
      <owl:onProperty rdf:resource='#PolicyAggregation'/>
      <owl:allValuesFrom rdf:resource='#Policy'/>
   </owl:Restriction>
   <rdfs:comment>Asociacion PolicyInSystem. 0..1</rdfs:comment>
   <owl:Restriction>
      <owl:onProperty rdf:resource='#PolicyInSystem'/>
      <owl:allValuesFrom rdf:resource='ccm:System'/>
   </owl:Restriction>
   <rdfs:comment>Asociacion ReusablePolicy. 0..1</rdfs:comment>
   <owl:Restriction>
      <owl:onProperty rdf:resource='#ReusablePolicy'/>
      <owl:allValuesFrom rdf:resource='#ReusablePolicyContainer'/>
   </owl:Restriction>
   <rdfs:comment>Asociacion PolicyActionStructure. 0..*</rdfs:comment>
   <owl:Restriction>
      <owl:onProperty rdf:resource='#PolicyActionStructure'/>
      <owl:allValuesFrom rdf:resource='#PolicyAction'/>
   </owl:Restriction>
</owl:Class>
<owl:DatatypeProperty rdf:ID='CommonName'>
   <rdfs:comment>DatatypeProperty para la propiedad CommonName</rdfs:comment>
   <rdf:type rdf:resource='owl:FunctionalProperty'/>
   <rdfs:range rdf:resource='xsd:string'/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID='PolicyKeyword'>
   <rdfs:comment>DatatypeProperty para la propiedad PolicyKeywords. Lista</rdfs:comment>
   <rdfs:range rdf:resource='xsd:string'/>
</owl:DatatypeProperty>
<owl:TransitiveProperty rdf:ID='PolicyComponent'>
   <rdfs:domain rdf:resource='#Policy'/>
   <rdfs:range rdf:resource='#Policy'/>
</owl:TransitiveProperty>
<owl:TransitiveProperty rdf:ID='PolicyAggregation'>
   <rdf:type rdf:resource='#AggregationProperty'/>
   <rdfs:domain rdf:resource='#Policy'/>
   <rdfs:range rdf:resource='#Policy'/>
   <owl:inverseOf rdf:resource='#PolicyComponent'/>
</owl:TransitiveProperty>
<owl:FunctionalProperty rdf:ID='PolicyInSystem'>
   <rdfs:comment>ObjectProperty para la asociacion PolicyInSystem</rdfs:comment>
   <rdfs:domain rdf:resource='#Policy'/>
   <rdfs:range rdf:resource='ccm:System'/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID='ReusablePolicy'>
   <rdfs:comment>ObjectProperty para la asociacion ReusablePolicy</rdfs:comment>
```

```
   <rdfs:domain rdf:resource='#Policy'/>
   <rdfs:range rdf:resource='#ReusablePolicyContainer'/>
</owl:FunctionalProperty>
<owl:Class rdf:about='ccm:System'>
   <rdfs:comment>Asociacion PolicyInSystem. 0..*</rdfs:comment>
   <owl:Restriction>
      <owl:onProperty rdf:resource='#SystemHasPolicy'/>
      <owl:allValuesFrom rdf:resource='#Policy'/>
   </owl:Restriction>
</owl:Class>
<owl:ObjectProperty rdf:ID='SystemHasPolicy'>
   <rdfs:domain rdf:resource='ccm:System'/>
   <rdfs:range rdf:resource='#Policy'/>
   <owl:inverseOf rdf:resource='#PolicyInSystem'/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID='PolicyActionStructure'>
   <rdf:type rdf:resource='#AggregationProperty'/>
   <rdfs:domain rdf:resource='#Policy'/>
   <rdfs:range rdf:resource='#PolicyAction'/>
   <owl:inverseOf rdf:resource='#PolicyActionInPolicy'/>
</owl:ObjectProperty>
<!-- ************************************ -->
<!--             Clase PolicySet              -->
<!-- ************************************ -->
<owl:Class rdf:ID='PolicySet'>
   <rdfs:comment>Clase PolicySet</rdfs:comment>
   <rdfs:label xml:lang='en'>policy-set</rdfs:label>
   <rdfs:label xml:lang='es'>conjunto-de-pol&acute;iticas</rdfs:label>
   <owl:functionalSubClassOf rdf:resource='#Policy'/>
</owl:Class>
<owl:Class rdf:about='PolicySet'>
   <rdfs:comment>Propiedad PolicyDecisionStrategy</rdfs:comment>
   <owl:Restriction>
      <owl:onProperty rdf:resource='#PolicyDecisionStrategy'/>
      <owl:cardinality>1</owl:cardinality>
   </owl:Restriction>
</owl:Class>
<owl:Class rdf:about='PolicySet'>
   <rdfs:comment>Propiedad PolicyRoles. Lista</rdfs:comment>
   <owl:Restriction>
      <owl:onProperty rdf:resource='#PolicyRole'/>
      <owl:minCardinality>1</owl:minCardinality>
   </owl:Restriction>
</owl:Class>
<owl:Class rdf:about='PolicySet'>
   <rdfs:comment>Asociacion PolicySetInSystem. 1..1</rdfs:comment>
   <owl:Restriction>
      <owl:onProperty rdf:resource='#PolicySetInSystem'/>
      <owl:allValuesFrom rdf:resource='ccm:System'/>
   </owl:Restriction>
   <owl:Restriction>
      <owl:onProperty rdf:resource='#PolicySetInSystem'/>
      <owl:cardinality>1</owl:cardinality>
   </owl:Restriction>
</owl:Class>
<owl:Class rdf:about='PolicySet'>
   <rdfs:comment>Asociacion PolicySetComponent. 0..*</rdfs:comment>
   <owl:Restriction>
      <owl:onProperty rdf:resource='#PolicySetComponent'/>
      <owl:allValuesFrom rdf:resource='#PolicySet'/>
   </owl:Restriction>
</owl:Class>
<owl:Class rdf:about='PolicySet'>
   <rdfs:comment>Asociacion PolicySetComponent. 0..*</rdfs:comment>
   <owl:Restriction>
      <owl:onProperty rdf:resource='#PolicySetAggregation'/>
      <owl:allValuesFrom rdf:resource='#PolicySet'/>
   </owl:Restriction>
</owl:Class>
<owl:DatatypeProperty rdf:ID='PolicyDecisionStrategy'>
```

```
   <rdfs:comment>DatatypeProperty para la propiedad PolicyDecisionStrategy</rdfs:comment>
   <rdf:type rdf:resource='owl:FunctionalProperty'/>
   <rdfs:range rdf:resource='xsd:uint16'/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID='PolicyRole'>
   <rdfs:comment>DatatypeProperty para la propiedad PolicyRoles. Lista</rdfs:comment>
   <rdfs:range rdf:resource='xsd:string'/>
</owl:DatatypeProperty>
<owl:TransitiveProperty rdf:ID='PolicySetComponent'>
   <rdfs:domain rdf:resource='#PolicySet'/>
   <rdfs:range rdf:resource='#PolicySet'/>
</owl:TransitiveProperty>
<owl:TransitiveProperty rdf:ID='PolicySetAggregation'>
   <rdf:type rdf:resource='#AggregationProperty'/>
   <rdfs:domain rdf:resource='#PolicySet'/>
   <rdfs:range rdf:resource='#PolicySet'/>
   <owl:inverseOf rdf:resource='#PolicySetComponent'/>
</owl:TransitiveProperty>
<owl:FunctionalProperty rdf:ID='PolicySetInSystem'>
   <rdfs:comment>ObjectProperty para la asociacion PolicySetInSystem</rdfs:comment>
   <rdfs:domain rdf:resource='#PolicySet'/>
   <rdfs:range rdf:resource='ccm:System'/>
</owl:FunctionalProperty>
<owl:Class rdf:about='ccm:System'>
   <rdfs:comment>Asociacion PolicySetInSystem. 0..*</rdfs:comment>
   <owl:Restriction>
      <owl:onProperty rdf:resource='#SystemHasPolicySet'/>
      <owl:allValuesFrom rdf:resource='#PolicySet'/>
   </owl:Restriction>
</owl:Class>
<owl:ObjectProperty rdf:ID='SystemHasPolicySet'>
   <rdfs:domain rdf:resource='ccm:System'/>
   <rdfs:range rdf:resource='#PolicySet'/>
   <owl:inverseOf rdf:resource='#PolicySetInSystem'/>
</owl:ObjectProperty>
<!-- ************************************ -->
<!--              Clase PolicyRule                -->
<!-- ************************************ -->
<owl:Class rdf:ID='PolicyRule'>
   <rdfs:comment>Clase PolicyRule</rdfs:comment>
   <rdfs:label xml:lang='en'>policy-rule</rdfs:label>
   <rdfs:label xml:lang='es'>regla</rdfs:label>
   <owl:functionalSubClassOf rdf:resource='#PolicySet'/>
</owl:Class>
<owl:Class rdf:about='PolicyRule'>
   <rdfs:comment>Propiedad CreationClassName, clave</rdfs:comment>
   <owl:Restriction>
      <owl:onProperty rdf:resource='#CreationClassName'/>
      <owl:cardinality>1</owl:cardinality>
   </owl:Restriction>
</owl:Class>
<owl:Class rdf:about='PolicyRule'>
   <rdfs:comment>Propiedad PolicyRuleName, clave</rdfs:comment>
   <owl:Restriction>
      <owl:onProperty rdf:resource='#PolicyRuleName'/>
      <owl:cardinality>1</owl:cardinality>
   </owl:Restriction>
</owl:Class>
<owl:Class rdf:about='PolicyRule'>
   <rdfs:comment>Propiedad Enabled</rdfs:comment>
   <owl:Restriction>
      <owl:onProperty rdf:resource='#Enabled'/>
      <owl:cardinality>1</owl:cardinality>
   </owl:Restriction>
</owl:Class>
<owl:Class rdf:about='PolicyRule'>
   <rdfs:comment>Propiedad ConditionListType</rdfs:comment>
   <owl:Restriction>
      <owl:onProperty rdf:resource='#ConditionListType'/>
      <owl:cardinality>1</owl:cardinality>
```

```
      </owl:Restriction>
   </owl:Class>
   <owl:Class rdf:about='PolicyRule'>
      <rdfs:comment>Propiedad RuleUsage</rdfs:comment>
      <owl:Restriction>
         <owl:onProperty rdf:resource='#RuleUsage'/>
         <owl:cardinality>1</owl:cardinality>
      </owl:Restriction>
   </owl:Class>
   <owl:Class rdf:about='PolicyRule'>
      <rdfs:comment>Propiedad Mandatory</rdfs:comment>
      <owl:Restriction>
         <owl:onProperty rdf:resource='#Mandatory'/>
         <owl:cardinality>1</owl:cardinality>
      </owl:Restriction>
   </owl:Class>
   <owl:Class rdf:about='PolicyRule'>
      <rdfs:comment>Propiedad SequencedActions</rdfs:comment>
      <owl:Restriction>
         <owl:onProperty rdf:resource='#SequencedActions'/>
         <owl:cardinality>1</owl:cardinality>
      </owl:Restriction>
   </owl:Class>
   <owl:Class rdf:about='PolicyRule'>
      <rdfs:comment>Propiedad ExecutionStrategy</rdfs:comment>
      <owl:Restriction>
         <owl:onProperty rdf:resource='#ExecutionStrategy'/>
         <owl:cardinality>1</owl:cardinality>
      </owl:Restriction>
   </owl:Class>
   <owl:Class rdf:about='PolicyRule'>
      <rdfs:comment>Asociacion PolicyRuleInSystem. 1..1</rdfs:comment>
      <owl:Restriction>
         <owl:onProperty rdf:resource='#PolicyRuleInSystem'/>
         <owl:allValuesFrom rdf:resource='ccm:System'/>
      </owl:Restriction>
      <owl:Restriction>
         <owl:onProperty rdf:resource='#PolicyRuleInSystem'/>
         <owl:cardinality>1</owl:cardinality>
      </owl:Restriction>
   </owl:Class>
   <!--
   <owl:Class rdf:about='PolicyRule'>
      <rdfs:comment>Asociacion PolicyRuleInPolicyGroup. 0..*</rdfs:comment>
      <owl:Restriction>
         <owl:onProperty rdf:resource='#PolicyRuleInPolicyGroup'/>
         <owl:allValuesFrom rdf:resource='#PolicyGroup'/>
      </owl:Restriction>
   </owl:Class>
-->
   <owl:Class rdf:about='PolicyRule'>
      <rdfs:comment>Asociacion PolicyRuleAggregatesPolicyCondition. 0..*</rdfs:comment>
      <owl:Restriction>
         <owl:onProperty rdf:resource='#PolicyRuleAggregatesPolicyCondition'/>
         <owl:allValuesFrom rdf:resource='#PolicyCondition'/>
      </owl:Restriction>
   </owl:Class>
   <owl:Class rdf:about='PolicyRule'>
      <rdfs:comment>Asociacion PolicyRuleInPolicyAction. 0..*</rdfs:comment>
      <owl:Restriction>
         <owl:onProperty rdf:resource='#PolicyRuleAggregatesPolicyAction'/>
         <owl:allValuesFrom rdf:resource='#PolicyAction'/>
      </owl:Restriction>
   </owl:Class>
   <owl:Class rdf:about='PolicyRule'>
      <rdfs:comment>Asociacion PolicyRuleValidityPeriod. 0..*</rdfs:comment>
      <owl:Restriction>
         <owl:onProperty rdf:resource='#PolicyRuleValidityPeriod'/>
         <owl:allValuesFrom rdf:resource='#PolicyTimePeriodCondition'/>
      </owl:Restriction>
```

```
   </owl:Class>
   <owl:DatatypeProperty rdf:ID='CreationClassName'>
      <rdfs:comment>DatatypeProperty para la propiedad CreationClassName. Clave</rdfs:comment>
      <rdf:type rdf:resource='owl:InverseFunctionalProperty'/>
      <rdfs:range rdf:resource='xsd:string'/>
   </owl:DatatypeProperty>
   <owl:DatatypeProperty rdf:ID='PolicyRuleName'>
      <rdfs:comment>DatatypeProperty para la propiedad PolicyRuleName. Clave</rdfs:comment>
      <rdf:type rdf:resource='owl:InverseFunctionalProperty'/>
      <rdfs:range rdf:resource='xsd:string'/>
   </owl:DatatypeProperty>
   <!-- ¿Como expresar que la clave es la union de las dos propiedades? -->
   <owl:DatatypeProperty rdf:ID='Enabled'>
      <rdfs:comment>DatatypeProperty para la propiedad Enabled</rdfs:comment>
      <rdf:type rdf:resource='owl:FunctionalProperty'/>
      <rdfs:range rdf:resource='xsd:uint16'/>
   </owl:DatatypeProperty>
   <owl:DatatypeProperty rdf:ID='ConditionListType'>
      <rdfs:comment>DatatypeProperty para la propiedad ConditionListType</rdfs:comment>
      <rdf:type rdf:resource='owl:FunctionalProperty'/>
      <rdfs:range rdf:resource='xsd:uint16'/>
   </owl:DatatypeProperty>
   <owl:DatatypeProperty rdf:ID='RuleUsage'>
      <rdfs:comment>DatatypeProperty para la propiedad RuleUsage</rdfs:comment>
      <rdf:type rdf:resource='owl:FunctionalProperty'/>
      <rdfs:range rdf:resource='xsd:string'/>
   </owl:DatatypeProperty>
   <owl:DatatypeProperty rdf:ID='Mandatory'>
      <rdfs:comment>DatatypeProperty para la propiedad Mandatory</rdfs:comment>
      <rdf:type rdf:resource='owl:FunctionalProperty'/>
      <rdfs:range rdf:resource='xsd:boolean'/>
   </owl:DatatypeProperty>
   <owl:DatatypeProperty rdf:ID='SequencedActions'>
      <rdfs:comment>DatatypeProperty para la propiedad SequencedActions</rdfs:comment>
      <rdf:type rdf:resource='owl:FunctionalProperty'/>
      <rdfs:range rdf:resource='xsd:uint16'/>
   </owl:DatatypeProperty>
   <owl:DatatypeProperty rdf:ID='ExecutionStrategy'>
      <rdfs:comment>DatatypeProperty para la propiedad ExecutionStrategy</rdfs:comment>
      <rdf:type rdf:resource='owl:FunctionalProperty'/>
      <rdfs:range rdf:resource='xsd:uint16'/>
   </owl:DatatypeProperty>
   <owl:FunctionalProperty rdf:ID='PolicyRuleInSystem'>
      <rdfs:comment>ObjectProperty para la asociacion PolicyRuleInSystem</rdfs:comment>
      <rdfs:domain rdf:resource='#PolicyRule'/>
      <rdfs:range rdf:resource='ccm:System'/>
   </owl:FunctionalProperty>
   <!--
   <owl:ObjectProperty rdf:ID='PolicyRuleInPolicyGroup'>
      <rdfs:domain rdf:resource='#PolicyRule'/>
      <rdfs:range rdf:resource='#PolicyGroup'/>
   </owl:ObjectProperty>
-->
   <owl:Class rdf:about='ccm:System'>
      <rdfs:comment>Asociacion PolicyRuleInSystem. 0..*, Weak</rdfs:comment>
      <owl:Restriction>
         <owl:onProperty rdf:resource='#SystemHasPolicyRule'/>
         <owl:allValuesFrom rdf:resource='#PolicyRule'/>
      </owl:Restriction>
   </owl:Class>
   <owl:ObjectProperty rdf:ID='SystemHasPolicyRule'>
      <rdf:type rdf:resource='#WeakProperty'/>
      <rdfs:domain rdf:resource='ccm:System'/>
      <rdfs:range rdf:resource='#PolicyRule'/>
      <owl:inverseOf rdf:resource='#PolicyRuleInSystem'/>
   </owl:ObjectProperty>
   <owl:ObjectProperty rdf:ID='PolicyRuleAggregatesPolicyCondition'>
      <rdf:type rdf:resource='#AggregationProperty'/>
      <rdfs:domain rdf:resource='#PolicyRule'/>
      <rdfs:range rdf:resource='#PolicyCondition'/>
```

```
   <owl:inverseOf rdf:resource='#PolicyConditionInPolicyRule'/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID='PolicyRuleAggregatesPolicyAction'>
   <rdf:type rdf:resource='#AggregationProperty'/>
   <rdfs:domain rdf:resource='#PolicyRule'/>
   <rdfs:range rdf:resource='#PolicyAction'/>
   <owl:inverseOf rdf:resource='#PolicyActionInPolicyRule'/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID='PolicyRuleValidityPeriod'>
   <rdf:type rdf:resource='#AggregationProperty'/>
   <rdfs:domain rdf:resource='#PolicyRule'/>
   <rdfs:range rdf:resource='#PolicyTimePeriodCondition'/>
   <owl:inverseOf rdf:resource='#ValidityPeriodInPolicyRule'/>
</owl:ObjectProperty>
<!-- ************************************ -->
<!--              Clase PolicyGroup               -->
<!-- ************************************ -->
<owl:Class rdf:ID='PolicyGroup'>
   <rdfs:comment>Clase PolicyGroup</rdfs:comment>
   <rdfs:label xml:lang='en'>policy-group</rdfs:label>
   <rdfs:label xml:lang='es'>grupo-de-politicas</rdfs:label>
   <owl:functionalSubClassOf rdf:resource='#PolicySet'/>
</owl:Class>
<owl:Class rdf:about='PolicyGroup'>
   <rdfs:comment>Propiedad CreationClassName, clave</rdfs:comment>
   <owl:Restriction>
      <owl:onProperty rdf:resource='#CreationClassName'/>
      <owl:cardinality>1</owl:cardinality>
   </owl:Restriction>
</owl:Class>
<owl:Class rdf:about='PolicyGroup'>
   <rdfs:comment>Propiedad PolicyGroupName. Clave</rdfs:comment>
   <owl:Restriction>
      <owl:onProperty rdf:resource='#PolicyGroupName'/>
      <owl:cardinality>1</owl:cardinality>
   </owl:Restriction>
</owl:Class>
<owl:Class rdf:about='PolicyGroup'>
   <rdfs:comment>Asociacion PolicyGroupInSystem. 1..1</rdfs:comment>
   <owl:Restriction>
      <owl:onProperty rdf:resource='#PolicyGroupInSystem'/>
      <owl:allValuesFrom rdf:resource='ccm:System'/>
   </owl:Restriction>
   <owl:Restriction>
      <owl:onProperty rdf:resource='#PolicyGroupInSystem'/>
      <owl:cardinality>1</owl:cardinality>
   </owl:Restriction>
</owl:Class>
<owl:Class rdf:about='PolicyGroup'>
   <rdfs:comment>Asociacion PolicyGroupInPolicyGroup. 0..*</rdfs:comment>
   <owl:Restriction>
      <owl:onProperty rdf:resource='#PolicyGroupInPolicyGroup'/>
      <owl:allValuesFrom rdf:resource='ccm:PolicyGroup'/>
   </owl:Restriction>
</owl:Class>
<!--
<owl:Class rdf:about='PolicyGroup'>
   <rdfs:comment>Asociacion PolicyGroupAggregatesPolicyGroup. 0..*</rdfs:comment>
   <owl:Restriction>
      <owl:onProperty rdf:resource='#PolicyGroupAggregatesPolicyGroup'/>
      <owl:allValuesFrom rdf:resource='ccm:PolicyGroup'/>
   </owl:Restriction>
</owl:Class>
<owl:Class rdf:about='PolicyGroup'>
   <rdfs:comment>Asociacion PolicyGroupAggregatesPolicyRule. 0..*</rdfs:comment>
   <owl:Restriction>
      <owl:onProperty rdf:resource='#PolicyGroupAggregatesPolicyRule'/>
      <owl:allValuesFrom rdf:resource='#PolicyRule'/>
   </owl:Restriction>
</owl:Class>
```

```
-->
   <owl:DatatypeProperty rdf:ID='PolicyGroupName'>
      <rdfs:comment>DatatypeProperty para la propiedad PolicyGroupName. Clave</rdfs:comment>
      <rdf:type rdf:resource='owl:InverseFunctionalProperty'/>
      <rdfs:range rdf:resource='xsd:string'/>
   </owl:DatatypeProperty>
   <owl:FunctionalProperty rdf:ID='PolicyGroupInSystem'>
      <rdfs:comment>ObjectProperty para la asociacion PolicyGroupInSystem</rdfs:comment>
      <rdfs:domain rdf:resource='#PolicyGroup'/>
      <rdfs:range rdf:resource='ccm:System'/>
   </owl:FunctionalProperty>
   <!--
   <owl:ObjectProperty rdf:ID='PolicyGroupAggregatesPolicyRule'>
      <rdf:type rdf:resource='#AggregationProperty'/>
      <rdfs:domain rdf:resource='#PolicyGroup'/>
      <rdfs:range rdf:resource='#PolicyRule'/>
      <owl:inverseOf rdf:resource='#PolicyRuleInPolicyGroup'/>
   </owl:ObjectProperty>
   <owl:TransitiveProperty rdf:ID='PolicyGroupInPolicyGroup'>
      <rdfs:domain rdf:resource='#PolicyGroup'/>
      <rdfs:range rdf:resource='#PolicyGroup'/>
   </owl:TransitiveProperty>
   <owl:TransitiveProperty rdf:ID='PolicyGroupAggregatesPolicyGroup'>
      <rdf:type rdf:resource='#AggregationProperty'/>
      <rdfs:domain rdf:resource='#PolicyGroup'/>
      <rdfs:range rdf:resource='#PolicyGroup'/>
      <owl:inverseOf rdf:resource='#PolicyGroupInPolicyGroup'/>
   </owl:TransitiveProperty>
-->
   <owl:Class rdf:about='ccm:System'>
      <rdfs:comment>Asociacion PolicyGroupInSystem. 0..*, Weak</rdfs:comment>
      <owl:Restriction>
         <owl:onProperty rdf:resource='#SystemHasPolicyGroup'/>
         <owl:allValuesFrom rdf:resource='#PolicyGroup'/>
      </owl:Restriction>
   </owl:Class>
   <owl:ObjectProperty rdf:ID='SystemHasPolicyGroup'>
      <rdf:type rdf:resource='#WeakProperty'/>
      <rdfs:domain rdf:resource='ccm:System'/>
      <rdfs:range rdf:resource='#PolicyGroup'/>
      <owl:inverseOf rdf:resource='#PolicyGroupInSystem'/>
   </owl:ObjectProperty>
   <!-- ************************************* -->
   <!--    Clase PolicyTimePeriodCondition    -->
   <!-- ************************************* -->
   <owl:Class rdf:ID='PolicyTimePeriodCondition'>
      <rdfs:comment>Clase PolicyTimePeriodCondition</rdfs:comment>
      <rdfs:label xml:lang='en'>policy-time-period-condition</rdfs:label>
      <rdfs:label xml:lang='es'>condici&acute;on-periodo-de-tiempo</rdfs:label>
      <owl:functionalSubClassOf rdf:resource='#PolicyCondition'/>
   </owl:Class>
   <owl:Class rdf:about='PolicyTimePeriodCondition'>
      <rdfs:comment>Propiedad TimePeriod</rdfs:comment>
      <owl:Restriction>
         <owl:onProperty rdf:resource='#TimePeriod'/>
         <owl:cardinality>1</owl:cardinality>
      </owl:Restriction>
   </owl:Class>
   <owl:Class rdf:about='PolicyTimePeriodCondition'>
      <rdfs:comment>Propiedad MonthOfYearMask</rdfs:comment>
      <owl:Restriction>
         <owl:onProperty rdf:resource='#MonthOfYearMask'/>
         <owl:cardinality>1</owl:cardinality>
      </owl:Restriction>
   </owl:Class>
   <owl:Class rdf:about='PolicyTimePeriodCondition'>
      <rdfs:comment>Propiedad DayOfMonthMask</rdfs:comment>
      <owl:Restriction>
         <owl:onProperty rdf:resource='#DayOfMonthMask'/>
         <owl:cardinality>1</owl:cardinality>
```

```
   </owl:Restriction>
</owl:Class>
<owl:Class rdf:about='PolicyTimePeriodCondition'>
   <rdfs:comment>Propiedad DayOfWeekMask</rdfs:comment>
   <owl:Restriction>
      <owl:onProperty rdf:resource='#DayOfWeekMask'/>
      <owl:cardinality>1</owl:cardinality>
   </owl:Restriction>
</owl:Class>
<owl:Class rdf:about='PolicyTimePeriodCondition'>
   <rdfs:comment>Propiedad TimeOfDayMask</rdfs:comment>
   <owl:Restriction>
      <owl:onProperty rdf:resource='#TimeOfDayMask'/>
      <owl:cardinality>1</owl:cardinality>
   </owl:Restriction>
</owl:Class>
<owl:Class rdf:about='PolicyTimePeriodCondition'>
   <rdfs:comment>Propiedad LocalOrUtcTime</rdfs:comment>
   <owl:Restriction>
      <owl:onProperty rdf:resource='#LocalOrUtcTime'/>
      <owl:cardinality>1</owl:cardinality>
   </owl:Restriction>
</owl:Class>
<owl:Class rdf:about='PolicyTimePeriodCondition'>
   <rdfs:comment>Asociacion ValidityPeriodInPolicyRule. 0..*</rdfs:comment>
   <owl:Restriction>
      <owl:onProperty rdf:resource='#ValidityPeriodInPolicyRule'/>
      <owl:allValuesFrom rdf:resource='#PolicyRule'/>
   </owl:Restriction>
</owl:Class>
<owl:ObjectProperty rdf:ID='ValidityPeriodInPolicyRule'>
   <rdfs:domain rdf:resource='#PolicyTimePeriodCondition'/>
   <rdfs:range rdf:resource='#PolicyRule'/>
</owl:ObjectProperty>
<!-- ************************************ -->
<!--      Clase VendorPolicyCondition     -->
<!-- ************************************ -->
<owl:Class rdf:ID='VendorPolicyCondition'>
   <rdfs:comment>Clase VendorPolicyCondition</rdfs:comment>
   <rdfs:label xml:lang='en'>vendor-policy-condition</rdfs:label>
   <rdfs:label xml:lang='es'>condici&acute;on-de-fabricante</rdfs:label>
   <owl:functionalSubClassOf rdf:resource='#PolicyCondition'/>
</owl:Class>
<owl:Class rdf:about='VendorPolicyCondition'>
   <rdfs:comment>Propiedad Constraint</rdfs:comment>
   <owl:Restriction>
      <owl:onProperty rdf:resource='#Constraint'/>
      <owl:cardinality>1</owl:cardinality>
   </owl:Restriction>
</owl:Class>
<owl:Class rdf:about='VendorPolicyCondition'>
   <rdfs:comment>Propiedad ConstraintEncoding</rdfs:comment>
   <owl:Restriction>
      <owl:onProperty rdf:resource='#ConstraintEncoding'/>
      <owl:cardinality>1</owl:cardinality>
   </owl:Restriction>
</owl:Class>
<!-- ************************************ -->
<!--            Clase PolicyCondition            -->
<!-- ************************************ -->
<owl:Class rdf:ID='PolicyCondition'>
   <rdfs:comment>Clase PolicyCondition</rdfs:comment>
   <rdfs:label xml:lang='en'>policy-condition</rdfs:label>
   <rdfs:label xml:lang='es'>condici&acute;on</rdfs:label>
   <owl:functionalSubClassOf rdf:resource='#Policy'/>
</owl:Class>
<owl:Class rdf:about='PolicyCondition'>
   <rdfs:comment>Propiedad SystemCreationClassName</rdfs:comment>
   <owl:Restriction>
      <owl:onProperty rdf:resource='#SystemCreationClassName'/>
```

```
      <owl:cardinality>1</owl:cardinality>
   </owl:Restriction>
</owl:Class>
<owl:Class rdf:about='PolicyCondition'>
   <rdfs:comment>Propiedad SystemName</rdfs:comment>
   <owl:Restriction>
      <owl:onProperty rdf:resource='#SystemName'/>
      <owl:cardinality>1</owl:cardinality>
   </owl:Restriction>
</owl:Class>
<owl:Class rdf:about='PolicyCondition'>
   <rdfs:comment>Propiedad PolicyRuleCreationClassName</rdfs:comment>
   <owl:Restriction>
      <owl:onProperty rdf:resource='#PolicyRuleCreationClassName'/>
      <owl:cardinality>1</owl:cardinality>
   </owl:Restriction>
</owl:Class>
<owl:Class rdf:about='PolicyCondition'>
   <rdfs:comment>Propiedad PolicyRuleName</rdfs:comment>
   <owl:Restriction>
      <owl:onProperty rdf:resource='#PolicyRuleName'/>
      <owl:cardinality>1</owl:cardinality>
   </owl:Restriction>
</owl:Class>
<owl:Class rdf:about='PolicyCondition'>
   <rdfs:comment>Propiedad CreationClassName</rdfs:comment>
   <owl:Restriction>
      <owl:onProperty rdf:resource='#CreationClassName'/>
      <owl:cardinality>1</owl:cardinality>
   </owl:Restriction>
</owl:Class>
<owl:Class rdf:about='PolicyCondition'>
   <rdfs:comment>Propiedad PolicyConditionName</rdfs:comment>
   <owl:Restriction>
      <owl:onProperty rdf:resource='#PolicyConditionName'/>
      <owl:cardinality>1</owl:cardinality>
   </owl:Restriction>
</owl:Class>
<owl:Class rdf:about='PolicyCondition'>
   <rdfs:comment>Asociacion PolicyConditionInPolicyRule. 0..*</rdfs:comment>
   <owl:Restriction>
      <owl:onProperty rdf:resource='#PolicyConditionInPolicyRule'/>
      <owl:allValuesFrom rdf:resource='#PolicyRule'/>
   </owl:Restriction>
</owl:Class>
<owl:ObjectProperty rdf:ID='PolicyConditionInPolicyRule'>
   <rdfs:domain rdf:resource='#PolicyCondition'/>
   <rdfs:range rdf:resource='#PolicyRule'/>
</owl:ObjectProperty>
<!-- ************************************* -->
<!--  Clase RuleSpecificPolicyCondition    -->
<!-- ************************************* -->
<owl:Class rdf:ID='RuleSpecificPolicyCondition'>
   <rdfs:comment>Clase RuleSpecificPolicyCondition</rdfs:comment>
   <rdfs:label xml:lang='en'>rule-specific-policy-condition</rdfs:label>
   <rdfs:label xml:lang='es'>condici&acute;on-espec&acut;ifica-de-una-regla</rdfs:label>
   <owl:functionalSubClassOf rdf:resource='#PolicyCondition'/>
</owl:Class>
<owl:Class rdf:about='RuleSpecificPolicyCondition'>
   <rdfs:comment>Asociacion PolicyConditionInPolicyRule. 0..*</rdfs:comment>
   <owl:Restriction>
      <owl:onProperty rdf:resource='#PolicyConditionInPolicyRule'/>
      <owl:allValuesFrom rdf:resource='#PolicyRule'/>
   </owl:Restriction>
   <owl:Restriction>
      <owl:onProperty rdf:resource='#PolicyConditionInPolicyRule'/>
      <owl:cardinality>1</owl:cardinality>
   </owl:Restriction>
</owl:Class>
<!-- ************************************* -->
```

```
<!--     Clase ReusablePolicyCondition      -->
<!-- ************************************* -->
<owl:Class rdf:ID='ReusablePolicyCondition'>
   <rdfs:comment>Clase ReusablePolicyCondition</rdfs:comment>
   <rdfs:label xml:lang='en'>reusable-policy-condition</rdfs:label>
   <rdfs:label xml:lang='es'>condici&acute;on-reutilizable</rdfs:label>
   <owl:functionalSubClassOf rdf:resource='#PolicyCondition'/>
</owl:Class>
<!-- ************************************* -->
<!--           Clase PolicyAction            -->
<!-- ************************************* -->
<owl:Class rdf:ID='PolicyAction'>
   <rdfs:comment>Clase PolicyAction</rdfs:comment>
   <rdfs:label xml:lang='en'>policy-action</rdfs:label>
   <rdfs:label xml:lang='es'>acci&acute;on</rdfs:label>
   <owl:functionalSubClassOf rdf:resource='#Policy'/>
</owl:Class>
<owl:Class rdf:about='PolicyAction'>
   <rdfs:comment>Propiedad SystemCreationClassName</rdfs:comment>
   <owl:Restriction>
      <owl:onProperty rdf:resource='#SystemCreationClassName'/>
      <owl:cardinality>1</owl:cardinality>
   </owl:Restriction>
</owl:Class>
<owl:Class rdf:about='PolicyAction'>
   <rdfs:comment>Propiedad SystemName</rdfs:comment>
   <owl:Restriction>
      <owl:onProperty rdf:resource='#SystemName'/>
      <owl:cardinality>1</owl:cardinality>
   </owl:Restriction>
</owl:Class>
<owl:Class rdf:about='PolicyAction'>
   <rdfs:comment>Propiedad PolicyRuleCreationClassName</rdfs:comment>
   <owl:Restriction>
      <owl:onProperty rdf:resource='#PolicyRuleCreationClassName'/>
      <owl:cardinality>1</owl:cardinality>
   </owl:Restriction>
</owl:Class>
<owl:Class rdf:about='PolicyAction'>
   <rdfs:comment>Propiedad PolicyRuleName</rdfs:comment>
   <owl:Restriction>
      <owl:onProperty rdf:resource='#PolicyRuleName'/>
      <owl:cardinality>1</owl:cardinality>
   </owl:Restriction>
</owl:Class>
<owl:Class rdf:about='PolicyAction'>
   <rdfs:comment>Propiedad CreationClassName</rdfs:comment>
   <owl:Restriction>
      <owl:onProperty rdf:resource='#CreationClassName'/>
      <owl:cardinality>1</owl:cardinality>
   </owl:Restriction>
</owl:Class>
<owl:Class rdf:about='PolicyAction'>
   <rdfs:comment>Propiedad PolicyActionName</rdfs:comment>
   <owl:Restriction>
      <owl:onProperty rdf:resource='#PolicyActionName'/>
      <owl:cardinality>1</owl:cardinality>
   </owl:Restriction>
</owl:Class>
<owl:Class rdf:about='PolicyAction'>
   <rdfs:comment>Asociacion PolicyActionInPolicyAction. 0..*</rdfs:comment>
   <owl:Restriction>
      <owl:onProperty rdf:resource='#PolicyActionInPolicyAction'/>
      <owl:allValuesFrom rdf:resource='#CompoundPolicyAction'/>
   </owl:Restriction>
</owl:Class>
<owl:Class rdf:about='PolicyAction'>
   <rdfs:comment>Asociacion PolicyActionInPolicyRule. 0..*</rdfs:comment>
   <owl:Restriction>
      <owl:onProperty rdf:resource='#PolicyActionInPolicyRule'/>
```

```
      <owl:allValuesFrom rdf:resource='#PolicyRule'/>
   </owl:Restriction>
</owl:Class>
<owl:ObjectProperty rdf:ID='PolicyActionInPolicyRule'>
   <rdfs:domain rdf:resource='#PolicyAction'/>
   <rdfs:range rdf:resource='#PolicyRule'/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID='PolicyActionInPolicyAction'>
   <rdfs:domain rdf:resource='#PolicyAction'/>
   <rdfs:range rdf:resource='#PolicyAction'/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID='PolicyActionInPolicy'>
   <rdfs:domain rdf:resource='#PolicyAction'/>
   <rdfs:range rdf:resource='#Policy'/>
</owl:ObjectProperty>
<!-- ************************************* -->
<!--      Clase VendorPolicyAction         -->
<!-- ************************************* -->
<owl:Class rdf:ID='VendorPolicyAction'>
   <rdfs:comment>Clase VendorPolicyAction</rdfs:comment>
   <rdfs:label xml:lang='en'>vendor-policy-action</rdfs:label>
   <rdfs:label xml:lang='es'>acci&acute;on-de-fabricante</rdfs:label>
   <owl:functionalSubClassOf rdf:resource='#PolicyAction'/>
</owl:Class>
<owl:Class rdf:about='VendorPolicyAction'>
   <rdfs:comment>Propiedad ActionData</rdfs:comment>
   <owl:Restriction>
      <owl:onProperty rdf:resource='#ActionData'/>
      <owl:cardinality>1</owl:cardinality>
   </owl:Restriction>
</owl:Class>
<owl:Class rdf:about='VendorPolicyAction'>
   <rdfs:comment>Propiedad ActionEncoding</rdfs:comment>
   <owl:Restriction>
      <owl:onProperty rdf:resource='#ActionEncoding'/>
      <owl:cardinality>1</owl:cardinality>
   </owl:Restriction>
</owl:Class>
<!-- ************************************* -->
<!--        Clase CompoundPolicyAction     -->
<!-- ************************************* -->
<owl:Class rdf:ID='CompoundPolicyAction'>
   <rdfs:comment>Clase CompoundPolicyAction</rdfs:comment>
   <rdfs:label xml:lang='en'>compound-policy-action</rdfs:label>
   <rdfs:label xml:lang='es'>condici&acute;on-compuesta</rdfs:label>
   <owl:functionalSubClassOf rdf:resource='#PolicyAction'/>
</owl:Class>
<owl:Class rdf:about='CompoundPolicyAction'>
   <rdfs:comment>Propiedad SequencedActions</rdfs:comment>
   <owl:Restriction>
      <owl:onProperty rdf:resource='#SequencedActions'/>
      <owl:cardinality>1</owl:cardinality>
   </owl:Restriction>
</owl:Class>
<owl:Class rdf:about='CompoundPolicyAction'>
   <rdfs:comment>Propiedad ExecutionStrategy</rdfs:comment>
   <owl:Restriction>
      <owl:onProperty rdf:resource='#SequencedStrategy'/>
      <owl:cardinality>1</owl:cardinality>
   </owl:Restriction>
</owl:Class>
<owl:Class rdf:about='CompoundPolicyAction'>
   <rdfs:comment>Asociacion PolicyActionInPolicyAction. 0..*</rdfs:comment>
   <owl:Restriction>
      <owl:onProperty rdf:resource='#PolicyActionAggregatesPolicyAction'/>
      <owl:allValuesFrom rdf:resource='#PolicyAction'/>
   </owl:Restriction>
</owl:Class>
<owl:ObjectProperty rdf:ID='PolicyActionAggregatesPolicyAction'>
   <rdfs:domain rdf:resource='#PolicyAction'/>
```

```
   <rdfs:range rdf:resource='#PolicyAction'/>
</owl:ObjectProperty>
<!-- ************************************* -->
<!--      Clase ReusablePolicyContainer      -->
<!-- ************************************* -->
<owl:Class rdf:ID='ReusablePolicyContainer'>
   <rdfs:comment>Clase ReusablePolicyContainer</rdfs:comment>
   <rdfs:label xml:lang='en'>reusable-policy-container</rdfs:label>
   <rdfs:label xml:lang='es'>contenedor-de-pol&acute;iticas-reutilizables</rdfs:label>
   <owl:functionalSubClassOf rdf:resource='ccm:AdminDomain'/>
</owl:Class>
<owl:ObjectProperty rdf:ID='ContainsReusablePolicy'>
   <rdfs:domain rdf:resource='#ReusablePolicyContainer'/>
   <rdfs:range rdf:resource='#Policy'/>
   <owl:inverseOf rdf:resource='ReusablePolicy'/>
</owl:ObjectProperty>
<!-- ********************************************************************** -->
<!--   Metapropiedades (Calificadores y Propiedades de Asociaciones)     -->
<!-- ********************************************************************** -->
<owl:ObjectProperty rdf:about='PolicyConditionInPolicyRule'>
   <owl:Restriction>
      <owl:onProperty rdf:resource='#GroupNumber'/>
      <owl:cardinality>1</owl:cardinality>
   </owl:Restriction>
   <owl:Restriction>
      <owl:onProperty rdf:resource='#ConditionNegated'/>
      <owl:cardinality>1</owl:cardinality>
   </owl:Restriction>
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:ID='GroupNumber'>
   <rdfs:comment>DatatypeProperty para la propiedad GroupNumber de la
                        asociacion PolicyConditionInPolicyRule</rdfs:comment>
   <rdf:type rdf:resource='owl:FunctionalProperty'/>
   <rdfs:range rdf:resource='xsd:uint16'/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID='ConditionNegated'>
   <rdfs:comment>DatatypeProperty para la propiedad ConditionNegated de la
                        asociacion PolicyConditionInPolicyRule</rdfs:comment>
   <rdf:type rdf:resource='owl:FunctionalProperty'/>
   <rdfs:range rdf:resource='xsd:boolean'/>
</owl:DatatypeProperty>
<owl:ObjectProperty rdf:about='PolicyActionStructure'>
   <owl:Restriction>
      <owl:onProperty rdf:resource='#ActionOrder'/>
      <owl:cardinality>1</owl:cardinality>
   </owl:Restriction>
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:ID='ActionOrder'>
   <rdfs:comment>DatatypeProperty para la propiedad ActionOrder de la
                        asociacion PolicyActionStructure</rdfs:comment>
   <rdf:type rdf:resource='owl:FunctionalProperty'/>
   <rdfs:range rdf:resource='xsd:uint16'/>
</owl:DatatypeProperty>
<owl:ObjectProperty rdf:about='PolicyActionInPolicyRule'>
   <owl:Restriction>
      <owl:onProperty rdf:resource='#ActionOrder'/>
      <owl:cardinality>1</owl:cardinality>
   </owl:Restriction>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about='PolicySetComponent'>
   <owl:Restriction>
      <owl:onProperty rdf:resource='#Priority'/>
      <owl:cardinality>1</owl:cardinality>
   </owl:Restriction>
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:ID='Priority'>
   <rdfs:comment>DatatypeProperty para la propiedad Priority de la
                        asociacion PolicySetComponent</rdfs:comment>
   <rdf:type rdf:resource='owl:FunctionalProperty'/>
   <rdfs:range rdf:resource='xsd:uint16'/>
```

```
   </owl:DatatypeProperty>
   <owl:ObjectProperty rdf:about='PolicySetInSystem'>
      <owl:Restriction>
         <owl:onProperty rdf:resource='#Priority'/>
         <owl:cardinality>1</owl:cardinality>
      </owl:Restriction>
   </owl:ObjectProperty>
   <!-- ************************************ -->
   <!--      Jerarquia de asociaciones       -->
   <!-- ************************************ -->
   <owl:ObjectProperty rdf:about='PolicyRuleInSystem'>
      <rdfs:subPropertyOf rdf:resource='PolicySetSystem'/>
   </owl:ObjectProperty>
   <owl:ObjectProperty rdf:about='PolicyGroupInSystem'>
      <rdfs:subPropertyOf rdf:resource='PolicySetSystem'/>
   </owl:ObjectProperty>
   <owl:ObjectProperty rdf:about='PolicySetInSystem'>
      <rdfs:subPropertyOf rdf:resource='PolicyInSystem'/>
   </owl:ObjectProperty>
   <owl:ObjectProperty rdf:about='ReusablePolicy'>
      <rdfs:subPropertyOf rdf:resource='PolicyInSystem'/>
   </owl:ObjectProperty>
   <owl:ObjectProperty rdf:about='PolicyInSystem'>
      <rdfs:subPropertyOf rdf:resource='ccm:Dependency'/>
   </owl:ObjectProperty>
</rdf:RDF>
```

A.4. Especificación $\mathcal{AL}\varepsilon\mathcal{CNOQ}^{-}_{\mathcal{H}R^{+}\circ}$ del Modelo de Políticas CIM

A continuación se describe la expresión en lógica descriptiva $\mathcal{AL}\varepsilon\mathcal{CNOQ}^{-}_{\mathcal{H}R^{+}\circ}$ del *Modelo de Políticas* de CIM recogido en las figuras A.3 y A.4.

$$
\begin{aligned}
Policy \sqsubseteq & Class \sqcap ManagedElement \sqcap \\
& \exists CommonName.string \sqcap \\
& \langle \leq 1 CommonName \rangle \sqcap \\
& \forall PolicyKeywords.string \sqcap \\
& \langle \geq 1 PolicyKeywords \rangle \sqcap \\
& \forall PolicyComponent\_Group^{-}.PolicyComponent \sqcap \\
& \forall PolicyComponent\_Component^{-}.PolicyComponent \sqcap \\
& \forall PolicyComponentRole.Policy \sqcap \\
& \forall PolicyInSystem\_Dependent^{-}.PolicyInSystem \sqcap \\
& \langle \leq 1 PolicyInSystem\_Dependent^{-} \rangle \sqcap \\
& \forall PolicyInSystemRole^{-}.System \sqcap \\
& \langle \leq 1 PolicyInSystemRole \rangle \sqcap \\
& \forall ReusablePolicy\_Dependent^{-}.ReusablePolicy \sqcap \\
& \langle \leq 1 ReusablePolicy\_Dependent^{-} \rangle \sqcap \\
& \forall ReusablePolicyRole^{-}.ReusablePolicyContainer \sqcap \\
& \langle \leq 1 ReusablePolicyRole \rangle \sqcap \\
& \forall PolicyActionStructure\_Group^{-}.PolicyActionStructure \sqcap \\
& \forall PolicyActionStructureRole.PolicyAction
\end{aligned}
$$

$$\begin{aligned} PolicyComponent \sqsubseteq & Aggregation \sqcap \\ & \exists PolicyComponent\_Part.Policy \sqcap \\ & \langle \leq 1 PolicyComponent\_Part \rangle \sqcap \\ & \exists PolicyComponent\_Group.Policy \sqcap \\ & \langle \leq 1 PolicyComponent\_Group \rangle \end{aligned}$$

$$PolicyComponentRole \equiv PolicyComponent\_Group^{-} \circ PolicyComponent\_Part$$

$$\begin{aligned} PolicyInSystem \sqsubseteq & Dependency \sqcap \\ & \exists PolicyInSystem\_Antecedent.System \sqcap \\ & \exists PolicyInSystem\_Dependent.Policy \sqcap \\ & \langle \leq 1 PolicyInSystem\_Antecedent \rangle \sqcap \\ & \langle \leq 1 PolicyInSystem\_Dependent \rangle \sqcap \end{aligned}$$

$$PolicyInSystemRole \equiv PolicyInSystem\_Antecedent^{-} \circ PolicyInSystem\_Dependent$$

$$\begin{aligned} ReusablePolicy \sqsubseteq & PolicyInSystem \sqcap \\ & \exists ReusablePolicy\_Antecedent.ReusablePolicyContainer \sqcap \\ & \exists ReusablePolicy\_Dependent.Policy \sqcap \\ & \langle \leq 1 ReusablePolicy\_Antecedent \rangle \sqcap \\ & \langle \leq 1 ReusablePolicy\_Dependent \rangle \sqcap \end{aligned}$$

$$ReusablePolicyRole \equiv ReusablePolicy\_Antecedent^{-} \circ ReusablePolicy\_Dependent$$

$$\begin{aligned} PolicyActionStructure \sqsubseteq & PolicyComponent \sqcap \\ & \exists ActionOrder.uint16 \\ & \langle \leq 1 ActionOrder \rangle \sqcap \\ & \exists PolicyActionStructure\_Group.Policy \sqcap \\ & \exists PolicyActionStructure\_Part.PolicyAction \sqcap \\ & \langle \leq 1 PolicyActionStructure\_Group \rangle \sqcap \\ & \langle \leq 1 PolicyActionStructure\_Part \rangle \end{aligned}$$

$$ActionOrder \sqsubseteq Property$$

$$\begin{aligned} PolicyActionStructureRole \equiv & PolicyActionStructure\_Group^{-} \circ \\ & PolicyActionStructure\_Part \end{aligned}$$

$$CommonName \sqsubseteq Property$$

$$PolicyKeywords \sqsubseteq Property$$

$$\exists PolicyInSystem.System \equiv PolicyInSystem : s$$

$$\begin{aligned} PolicySet \sqsubseteq & Class \sqcap Policy \sqcap \\ & \exists PolicyDecisionStrategy.uint16 \sqcap \\ & \langle \leq 1 PolicyDecisionStrategy \rangle \sqcap \\ & \forall PolicyRoles.string \sqcap \\ & \langle \geq 1 PolicyRoles \rangle \sqcap \\ & \forall PolicySetComponent\_Group^{-}.PolicySetComponent \sqcap \\ & \forall PolicySetComponent\_Part^{-}.PolicySetComponent \sqcap \\ & \forall PolicySetComponentRole.PolicySet \sqcap \\ & \exists PolicySetInSystem\_Dependent^{-}.PolicySetInSystem \sqcap \\ & \langle \leq 1 PolicySetInSystem\_Dependent^{(-)} \rangle \sqcap \\ & \exists PolicySetInSystemRole^{-}.System \sqcap \\ & \langle \leq 1 PolicySetInSystemRole^{-} \rangle \end{aligned}$$

$$\begin{aligned} PolicySetComponent \sqsubseteq & PolicyComponent \sqcap \\ & \exists Priority.uint16 \sqcap \\ & \langle \leq 1 Priority \rangle \sqcap \\ & \exists PolicySetComponent\_Group.PolicySet \sqcap \\ & \langle \leq 1 PolicySetComponent\_Group \rangle \sqcap \\ & \exists PolicySetComponent\_Part.PolicySet \sqcap \\ & \langle \leq 1 PolicySetComponent\_Part \rangle \sqcap \end{aligned}$$

$$PolicySetComponentRole \equiv PolicySetComponent\_Group^{-} \circ PolicySetComponent\_Part$$

$$Priority \sqsubseteq Property$$

$$\begin{aligned} PolicySetInSystem \sqsubseteq & PolicyInSystem \sqcap \\ & \exists Priority.uint16 \sqcap \\ & \exists PolicySetInSystem\_Dependent.PolicySet \sqcap \\ & \exists PolicySetInSystem\_Antecedent.System \sqcap \\ & \langle \leq 1 PolicySetInSystem\_Dependent \rangle \sqcap \\ & \langle \leq 1 PolicySetInSystem\_Antecedent \rangle \sqcap \end{aligned}$$

$$\begin{aligned} PolicySetInSystemRole \equiv & PolicySetInSystem\_Antecedent^{-} \circ \\ & PolicySetInSystem\_Dependent \end{aligned}$$

$$CreationClassName \sqsubseteq KeyProperty$$

$$PolicyGroupName \sqsubseteq KeyProperty$$

$$\begin{aligned}PolicyGroup \sqsubseteq & Class \sqcap PolicySet \sqcap \\ & \exists CreationClassName.string \sqcap \\ & \langle \leq 1 CreationClassName \rangle \sqcap \\ & \exists PolicyGroupName.string \sqcap \\ & \langle \leq 1 PolicyGroupName \rangle \sqcap \\ & \exists PolicyGroupInSystem\_Dependent^{-}.PolicyGroupInSystem \sqcap \\ & \langle \leq 1 PolicyGroupInSystem\_Dependent^{-} \rangle \sqcap \\ & \exists PolicyGroupInSystemRole^{-}.System \sqcap \\ & \langle \leq 1 PolicyGroupInSystemRole^{-} \rangle\end{aligned}$$

$$\begin{aligned}PolicyGroupInSystem \sqsubseteq & PolicySetInSystem \sqcap \\ & \exists PolicyGroupInSystem\_Dependent.PolicyGroup \sqcap \\ & \exists PolicyGroupInSystem\_Antecedent.System \sqcap \\ & \langle \leq 1 PolicyGroupInSystem\_Dependent \rangle \sqcap \\ & \langle \leq 1 PolicyGroupInSystem\_Antecedent \rangle \sqcap\end{aligned}$$

$$PolicyGroupInSystem\_Dependent \sqsubseteq WeakAssociation$$

$$\begin{aligned}PolicyGroupInSystemRole \equiv & PolicyGroupInSystem\_Antecedent^{-} \circ \\ & PolicyGroupInSystem\_Dependent\end{aligned}$$

$$PolicyRuleName \sqsubseteq KeyProperty$$

$$Enabled \sqsubseteq Property$$

$$ConditionListType \sqsubseteq Property$$

$$RuleUsage \sqsubseteq Property$$

$$Mandatory \sqsubseteq Property$$

$$SequencedActions \sqsubseteq Property$$

$$ExecutionStrategy \sqsubseteq Property$$

$$\begin{aligned}PolicyRuleInSystem \sqsubseteq & PolicySetInSystem \sqcap \\ & \exists PolicyRuleInSystem\_Dependent.PolicyRule \sqcap \\ & \exists PolicyRuleInSystem\_Antecedent.System \sqcap \\ & \langle \leq 1 PolicyRuleInSystem\_Dependent \rangle \sqcap \\ & \langle \leq 1 PolicyRuleInSystem\_Antecedent \rangle \sqcap\end{aligned}$$

$$PolicyRuleInSystem\_Dependent \sqsubseteq WeakAssociation$$

$$PolicyRuleInSystemRole \equiv PolicyRuleInSystem\_Group^{-} \circ PolicyRuleInSystem\_Part$$

$$
\begin{aligned}
PolicyRule \sqsubseteq & Class \sqcap PolicySet \sqcap \neg PolicyGroup \sqcap \\
& \exists CreationClassName.string \sqcap \\
& \langle \leq 1 CreationClassName \rangle \sqcap \\
& \exists PolicyRuleName.string \sqcap \\
& \langle \leq 1 PolicyRuleName \rangle \sqcap \\
& \exists Enabled.uint16 \sqcap \\
& \langle \leq 1 Enabled \rangle \sqcap \\
& \exists ConditionListType.uint16 \sqcap \\
& \langle \leq 1 ConditionListType \rangle \sqcap \\
& \exists RuleUsage.string \sqcap \\
& \langle \leq 1 RuleUsage \rangle \sqcap \\
& \exists Mandatory.boolean \sqcap \\
& \langle \leq 1 Mandatory \rangle \sqcap \\
& \exists SequencedActions.uint16 \sqcap \\
& \langle \leq 1 SequencedActions \rangle \sqcap \\
& \exists ExecutionStrategy.uint16 \sqcap \\
& \langle \leq 1 ExecutionStrategy \rangle \sqcap \\
& \exists PolicyRuleInSystem\_Dependent^{-}.PolicyRule \sqcap \\
& \langle \leq 1 PolicyRuleInSystem\_Dependent^{-} \rangle \sqcap \\
& \exists PolicyRuleInSystemRole.System \sqcap \\
& \langle \leq 1 PolicyRuleInSystemRole \rangle \\
& \forall PolicyRuleValidityPeriod\_Group^{-}.PolicyRuleValidityPeriod \sqcap \\
& \forall PolicyRuleValidityPeriodRole.PolicyTimePeriodCondition \\
& \forall PolicyActionInPolicyRule\_Group^{-}.PolicyActionInPolicyRule \sqcap \\
& \forall PolicyActionInPolicyRuleRole.PolicyAction \\
& \forall PolicyConditionInPolicyRule\_Group^{-}.PolicyConditionInPolicyRule \sqcap \\
& \forall PolicyConditionInPolicyRuleRole.PolicyCondition
\end{aligned}
$$

$$
\begin{aligned}
PolicyActionInPolicyRule \sqsubseteq & PolicyActionStructure \sqcap \\
& \exists PolicyActionInPolicyRule\_Group.PolicyRule \sqcap \\
& \exists PolicyActionInPolicyRule\_Part.PolicyAction \sqcap \\
& \langle \leq 1 PolicyActionInPolicyRule\_Group \rangle \sqcap \\
& \langle \leq 1 PolicyActionInPolicyRule\_Part \rangle
\end{aligned}
$$

$$
\begin{aligned}
PolicyActionInPolicyRuleRole \equiv & PolicyActionInPolicyRule\_Group^{-} \circ \\
& PolicyActionInPolicyRule\_Part
\end{aligned}
$$

$$
\begin{aligned}
PolicyRuleValidityPeriod \sqsubseteq & PolicyComponent \sqcap \\
& \exists PolicyRuleValidityPeriod\_Group.PolicyRule \sqcap \\
& \exists PolicyRuleVAlidityPeriod\_Part.PolicyTimePeriodCondition \sqcap \\
& \langle \leq 1 PolicyRuleValidityPeriod\_Group \rangle \sqcap \\
& \langle \leq 1 PolicyRuleValidityPeriod\_Part \rangle
\end{aligned}
$$

$$PolicyRuleValidityPeriodRole \equiv PolicyRuleValidityPeriod\_Group^{-} \circ PolicyRuleValidityPeriod\_Part$$

$$\begin{aligned}
PolicyAction \sqsubseteq\ & Class \sqcap Policy \sqcap \\
& \exists SystemCreationClassName.string \sqcap \\
& \langle \leq 1 SystemCreationClassName \rangle \sqcap \\
& \exists SystemName.string \sqcap \\
& \langle \leq 1 SystemName \rangle \sqcap \\
& \exists PolicyRuleCreationClassName.string \sqcap \\
& \langle \leq 1 PolicyRuleCreationClassName \rangle \sqcap \\
& \exists PolicyRuleName.string \sqcap \\
& \langle \leq 1 PolicyRuleName \rangle \sqcap \\
& \exists CreationClassName.string \sqcap \\
& \langle \leq 1 CreationClassName \rangle \sqcap \\
& \exists PolicyActionName.string \sqcap \\
& \langle \leq 1 PolicyActionName \rangle \sqcap \\
& \forall PolicyActionStructure_Part^{-}.PolicyAction \sqcap \\
& \forall PolicyActionStructureRole^{-}.Policy \sqcap \\
& \forall PolicyActionInPolicyAction_Part^{-}.PolicyActionInPolicyAction \sqcap \\
& \forall PolicyActionInPolicyActionRole^{-}.CompoundPolicyAction \\
& \forall PolicyActionInPolicyRule_Part^{-}.PolicyActionInPolicyRule \sqcap \\
& \forall PolicyActionInPolicyRuleRole^{-}.PolicyRule
\end{aligned}$$

$$SystemCreationClassName \sqsubseteq KeyProperty$$

$$SystemName \sqsubseteq KeyProperty$$

$$PolicyRuleCreationClassName \sqsubseteq KeyProperty$$

$$PolicyRuleName \sqsubseteq KeyProperty$$

$$CreationClassName \sqsubseteq KeyProperty$$

$$PolicyActionName \sqsubseteq KeyProperty$$

$$\begin{aligned}
CompoundPolicyAction \sqsubseteq\ & Class \sqcap PolicyAction \sqcap \\
& \exists SequencedActions.uint16 \sqcap \\
& \langle \leq 1 SequencedActions \rangle \sqcap \\
& \exists ExecutionStrategy.uint16 \sqcap \\
& \langle \leq 1 ExecutionStrategy \rangle \sqcap \\
& \forall PolicyActionInPolicyAction_Group^{-}.PolicyActionInPolicyAction \sqcap \\
& \forall PolicyActionInPolicyActionRole.PolicyAction
\end{aligned}$$

$$PolicyActionInPolicyAction \sqsubseteq PolicyActionStructure \sqcap \exists PolicyActionInPolicyAction\_Group.CompoundPolicyAction \sqcap \exists PolicyActionInPolicyAction\_Part.PolicyAction \sqcap \langle \leq 1 PolicyActionInPolicyAction\_Group \rangle \sqcap \langle \leq 1 PolicyActionInPolicyAction\_Part \rangle$$

$$PolicyActionInPolicyActionRole \equiv PolicyActionInPolicyAction\_Group^{-} \circ PolicyActionInPolicyAction\_Part$$

$$VendorPolicyAction \sqsubseteq Class \sqcap PolicyAction \sqcap \neg CompoundPolicyAction \sqcap \forall ActionData.Octetstring \sqcap \langle \geq 1 ActionData \rangle \sqcap \exists ActionEncoding.string \sqcap \langle \leq 1 ActionEncoding \rangle \sqcap$$

$$ActionEncoding \sqsubseteq Property \sqcap OIDProperty$$

$$PolicyCondition \sqsubseteq Class \sqcap Policy \sqcap \exists SystemCreationClassName.string \sqcap \langle \leq 1 SystemCreationClassName \rangle \sqcap \exists SystemName.string \sqcap \langle \leq 1 SystemName \rangle \sqcap \exists PolicyRuleCreationClassName.string \sqcap \langle \leq 1 PolicyRuleCreationClassName \rangle \sqcap \exists PolicyRuleName.string \sqcap \langle \leq 1 PolicyRuleName \rangle \sqcap \exists CreationClassName.string \sqcap \langle \leq 1 CreationClassName \rangle \sqcap \exists PolicyConditionName.string \sqcap \langle \leq 1 PolicyConditionName \rangle \sqcap \forall PolicyConditionInPolicyRule\_Part^{-}.PolicyConditionInPolicyRule \sqcap \forall PolicyConditionInPolicyRuleRole^{-}.PolicyRule$$

$$PolicyConditionName \sqsubseteq KeyProperty$$

$$VendorPolicyCondition \sqsubseteq Class \sqcap PolicyCondition \sqcap \forall Constraint.Octetstring \sqcap \langle \geq 1 Constraint \rangle \sqcap \exists ConstraintEncoding.string \sqcap \langle \leq 1 ConstraintEncoding \rangle \sqcap$$

$$ConstraintEncoding \sqsubseteq Property \sqcap OIDProperty$$

$$\begin{aligned}PolicyTimePeriodCondition \sqsubseteq & Class \sqcap PolicyCondition \sqcap \\ & \exists TimePeriod.string \sqcap \\ & \langle \leq 1 TimePeriod \rangle \sqcap \\ & \forall MonthOfYearMask.uint8 \sqcap \\ & \langle \geq 1 MonthOfYearMask \rangle \sqcap \\ & \forall DayOfMonthMask.uint8 \sqcap \\ & \langle \geq 1 DayOfMonthMask \rangle \sqcap \\ & \forall DayOfWeekMask.uint8 \sqcap \\ & \langle \geq 1 DayOfWeekMask \rangle \sqcap \\ & \exists TimeOfDayMask.string \sqcap \\ & \langle \leq 1 TimeOfDayMask \rangle \sqcap \\ & \exists LocalOrUtcTime.uint16 \sqcap \\ & \langle \leq 1 LocalOrUtcTime \rangle \sqcap \\ & \forall PolicyRuleValidityPeriod\_Part^{-}.PolicyRuleValidityPeriod \sqcap \\ & \forall PolicyRuleValidityPeriodRole^{-}.PolicyRule\end{aligned}$$

$$TimePeriod \sqsubseteq Property$$

$$MonthOfYearMask \sqsubseteq Property$$

$$DayOfMonthMask \sqsubseteq Property$$

$$DayOfWeekMask \sqsubseteq Property$$

$$TimeOfDayMask \sqsubseteq Property$$

$$LocalOrUtcTime \sqsubseteq Property$$

$$\begin{aligned}ReusablePolicyContainer \sqsubseteq & Class \sqcap AdminDomain \sqcap \\ & \forall ReusablePolicy\_Antecedent^{-}.ReusablePolicy \sqcap \\ & \forall ReusablePolicyRole.Policy\end{aligned}$$

Apéndice B

Semántica Formal del lenguaje ACSL

Índice General

B.1. Descripción UML de la sintaxis abstracta 243
B.2. Sintaxis abstracta 244
B.3. Sintaxis XML (XML Schema) 247

B.1. Descripción UML de la sintaxis abstracta

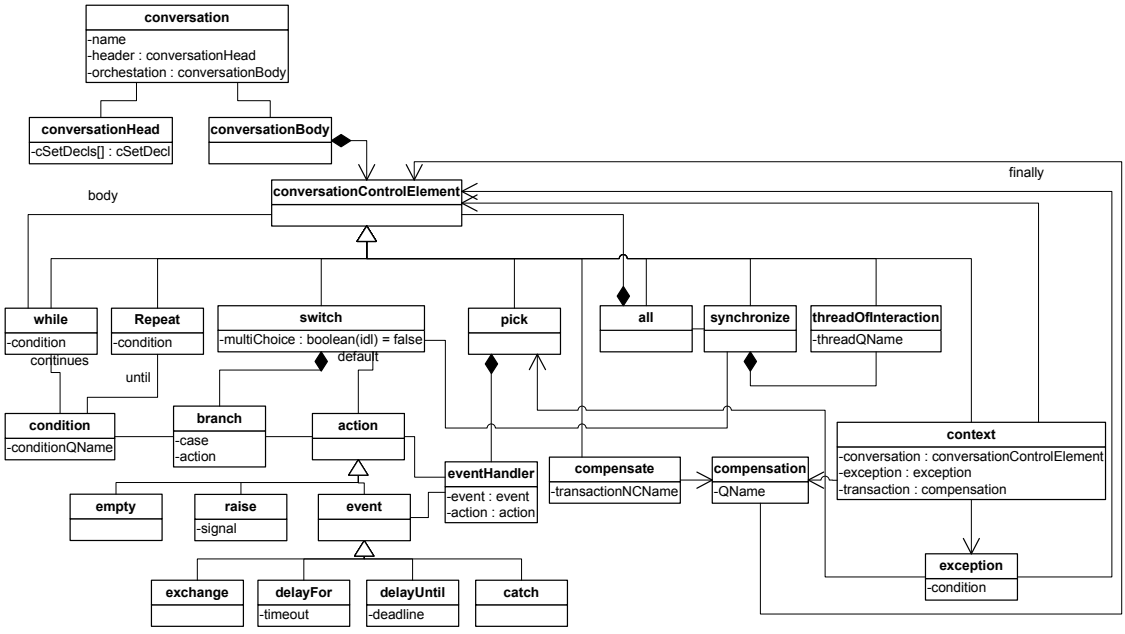

Figura B.1: Elementos de la sintaxis abstracta del lenguaje ACSL

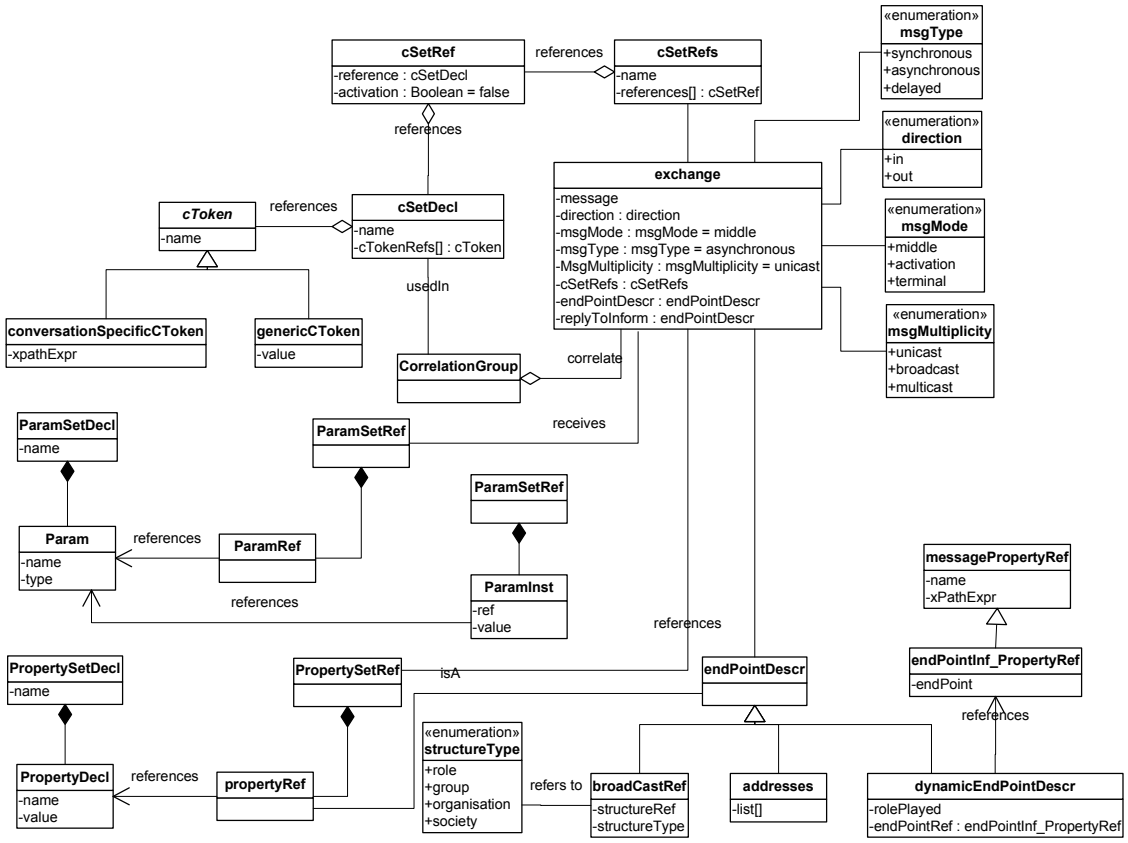

Figura B.2: Elementos ACSL para la descripción de los intercambios de mensajes correlacionados

B.2. Sintaxis abstracta

Se detalla a continuación la sintaxis abstracta del lenguaje ACSL. La gramática se describe en notación ABNF [IET97]. Las palabras reservadas del lenguaje aparecen en negrita y los no terminales comienzan por mayúscula.

Si bien esta notación permite describir la sintaxis abstracta ACSL de forma clara y próxima a la sintaxis XML que se detalla más adelante, debe resaltarse que la sintaxis abstracta ha sido definida para que pueda expresarse en una gramática LL(1), de modo que se facilite la realización de compiladores sencillos para otras sintaxis ACSL diferentes de la sintaxis XML.

```
Protocol := Name Body

Body := body Header Orchestation

Header := header {CSetDecl} [ParamSetDecl]

Orchestation := orchestation {ThreadOfInteraction}
```

```
ProtocolRef := protocolRef Ref ParamSetInst

ParamSetDecl := paramSetDecl Name {Param}

Param := Name Type Activation

Activation := activation boolean

ParamSetInst := paramSetInst {ParamInst}

ParamInst := paramInst Ref Value

ParamSetRef := paramSetRef {ParamRef}

ParamRef := paramRef Mode string

Mode := match | adjust

CSetDecl := Name cSetDecl {CTokenRef}

CTokenRef := cTokenRef string

CToken := cToken Name (ProtocolSpecificCToken | GenericCToken)

Name := name string

Ref := ref string

ProtocolSpecificCToken := messagePath

GenericCToken := value

CSetRef := cSetRef Activation {Reference}

Reference :=

CSetRefs := cSetRefs Name {CSetRef}

MessagePropertyRef := messageProperty

Exchange := exchange [ParamSetRef] [CSetRefs] [EndPointDescr]
[ReplyToInform] Message Direction Mode [Handling][Delivery]

Message := cfp | propose | inform | failure | request | ...

Handling := synchronous | asynchronous | delayed

Mode := middle | activation | terminal

Direction := in | out

Delivery := unreliable | reliable

Action := Exchange | Raise | DelayFor | DelayUntil
```

```
DelayFor := delayFor TimeOut

TimeOut := QName

DelayUntil := delayUntil Deadline

DeadLine := QName

Raise := raise Signal [ParamSetInst]

Signal := QName

ProtocolControlGroup := Empty |
                        ThreadOfInteraction |
                        ThreadOfInteractionRef |
                        ProtocolRef |
                        Switch |
                        All |
                        Pick |
                        While |
                        Context |
                        Compensate |
                        Synchronize

ThreadOfInteraction := threadOfInteraction [Name] [Activation]
[ParamSetDecl] {protocolControlGroup | Action}

ThreadOfInteractionRef := threadOfInteractionRef Ref
[ParamSetInst]

Synchronize := synchronize {ThreadOfInteractionRef}

All := all {ProtocolControlGroup} [Synchronize]

Pick := pick [Times] [ParamSetRef] {EventHandler} [OnTimes]

OnTimes := onTimes (ThreadOfInteraction | ProtocolControlGroup |
Action)

Times := Expression

EventHandler := eventHandler Event ActionBlock

ActionBlock := action (ThreadOfInteraction | ProtocolControlGroup
| Action)

Event := event Id (DelayFor | DelayUntil | Exchange | Catch)

Id:= id string

Switch := switch Multichoice {Branch} [Default]

Multichoice := multichoice boolean

Branch := branch Case ActionBlock
```

```
Case:= case Condition

Condition := condition [ParamSetRef] Expression

boolean := true | false

While := while Condition ActionBlock

Exception := exception [ParamSetDecl] Pick [Finally]

Finally := finally ProtocolControlGroup

Compensation := compensation Name (ThreadOfInteraction |
ProtocolControlGroup | Action)

Empty := empty

Context := context (ThreadOfInteraction | ProtocolControlGroup |
Action) [Exception] [Compensation]

Compensate := compensate Ref [ParamSetRef]
```

B.3. Sintaxis XML (XML Schema)

```
<xs:schema targetNamespace='urn:TIC2001-3451:schemas:acsl'
xmlns:xs='http://www.w3.org/2001/XMLSchema'
xmlns:mstns='urn:TIC2001-3451:schemas:acsl'
xmlns='urn:TIC2001-3451:schemas:acsl'
elementFormDefault='qualified'>
   <xs:complexType name='documentation' abstract='true'>
      <xs:sequence>
         <xs:element ref='annotation' minOccurs='0'/>
         <xs:element ref='appInfo' minOccurs='0'/>
      </xs:sequence>
   </xs:complexType>
   <xs:element name='annotation'>
      <xs:complexType mixed='true'>
         <xs:choice minOccurs='0' maxOccurs='unbounded'>
            <xs:any minOccurs='0' maxOccurs='unbounded'/>
         </xs:choice>
         <xs:anyAttribute/>
      </xs:complexType>
   </xs:element>
   <xs:element name='appInfo'>
      <xs:complexType mixed='true'>
         <xs:choice minOccurs='0' maxOccurs='unbounded'>
            <xs:any minOccurs='0' maxOccurs='unbounded'/>
         </xs:choice>
         <xs:anyAttribute/>
      </xs:complexType>
   </xs:element>
   <xs:complexType name='protocolRefType'>
      <xs:sequence>
         <xs:element name='description' type='xs:string' minOccurs='0'/>
         <xs:element name='paramSetInst' type='paramSetInstType' minOccurs='0'/>
      </xs:sequence>
      <xs:attribute name='ref' type='xs:QName'/>
   </xs:complexType>
   <xs:element name='protocolRef' type='protocolRefType'/>
   <xs:complexType name='paramType'>
      <xs:sequence>
         <xs:element name='name' type='xs:NCName'/>
         <xs:element name='type' type='xs:string'/>
```

```
      <xs:element name='activationValue' type='xs:string' minOccurs='0'/>
   </xs:sequence>
</xs:complexType>
<xs:complexType name='paramInstType'>
   <xs:sequence>
      <xs:element name='ref' type='xs:NCName'/>
      <xs:element name='value' type='xs:string'/>
   </xs:sequence>
</xs:complexType>
<xs:complexType name='paramSetDeclType'>
   <xs:sequence>
      <xs:element name='param' type='paramType' maxOccurs='unbounded'/>
   </xs:sequence>
   <xs:attribute name='name' type='xs:string' use='required'/>
</xs:complexType>
<xs:complexType name='paramSetInstType'>
   <xs:sequence>
      <xs:element name='paramInst' type='paramInstType' maxOccurs='unbounded'/>
   </xs:sequence>
</xs:complexType>
<xs:complexType name='paramSetRefType'>
   <xs:sequence>
      <xs:element name='paramRef' maxOccurs='unbounded'>
         <xs:complexType>
            <xs:simpleContent>
               <xs:extension base='xs:string'>
                  <xs:attribute name='mode'>
                     <xs:simpleType>
                        <xs:restriction base='xs:string'>
                           <xs:enumeration value='match'/>
                           <xs:enumeration value='adjust'/>
                        </xs:restriction>
                     </xs:simpleType>
                  </xs:attribute>
               </xs:extension>
            </xs:simpleContent>
         </xs:complexType>
      </xs:element>
   </xs:sequence>
</xs:complexType>
<xs:complexType name='cTokenType' abstract='true'>
   <xs:sequence>
      <xs:element name='name' type='xs:NCName'/>
   </xs:sequence>
</xs:complexType>
<xs:complexType name='protocolSpecificCTokenType'>
   <xs:complexContent>
      <xs:extension base='cTokenType'>
         <xs:sequence>
            <xs:element name='xPathExpr' type='xs:string'/>
         </xs:sequence>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
<xs:complexType name='genericCTokenType'>
   <xs:complexContent>
      <xs:extension base='cTokenType'>
         <xs:sequence>
            <xs:element name='value' type='xs:anyURI'/>
         </xs:sequence>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
<xs:complexType name='cSetDeclType'>
   <xs:sequence>
      <xs:element name='cTokenRef' type='xs:QName' maxOccurs='unbounded'/>
   </xs:sequence>
   <xs:attribute name='name' type='xs:NCName' use='required'/>
</xs:complexType>
<xs:complexType name='cSetRefType'>
```

```
   <xs:sequence>
      <xs:element name='reference' type='xs:QName' maxOccurs='unbounded'/>
   </xs:sequence>
   <xs:attribute name='activation' type='xs:boolean' default='false'/>
</xs:complexType>
<xs:complexType name='cSetRefsType'>
   <xs:sequence>
      <xs:element name='cSetRef' type='cSetRefType' maxOccurs='unbounded'/>
   </xs:sequence>
   <xs:attribute name='name' type='xs:NCName' use='required'/>
</xs:complexType>
<xs:complexType name='messagePropertyRefType'>
   <xs:sequence>
      <xs:element name='name' type='xs:NCName'/>
      <xs:element name='XPathExpr' type='xs:string'/>
   </xs:sequence>
</xs:complexType>
<xs:complexType name='endPointInf_PropertyRefType'>
   <xs:complexContent>
      <xs:extension base='messagePropertyRefType'>
         <xs:attribute name='endPoint' type='xs:QName'/>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
<xs:element name='dynamicEndPointRef'>
   <xs:complexType>
      <xs:sequence>
         <xs:element name='rolePlayed' type='xs:QName'/>
         <xs:element name='endPointRef' type='endPointInf_PropertyRefType'/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
<xs:element name='broadcastRef'>
   <xs:complexType>
      <xs:sequence>
         <xs:element name='structureRef' type='xs:QName'/>
         <xs:element name='structureType'>
            <xs:simpleType>
               <xs:restriction base='xs:string'>
                  <xs:enumeration value='role'/>
                  <xs:enumeration value='group'/>
                  <xs:enumeration value='organisation'/>
                  <xs:enumeration value='society'/>
               </xs:restriction>
            </xs:simpleType>
         </xs:element>
      </xs:sequence>
   </xs:complexType>
</xs:element>
<xs:complexType name='endPointDescrType'>
   <xs:choice>
      <xs:element ref='dynamicEndPointRef'/>
      <xs:element name='addresses'>
         <xs:simpleType>
            <xs:list itemType='xs:anyURI'/>
         </xs:simpleType>
      </xs:element>
      <xs:element ref='broadcastRef'/>
   </xs:choice>
   <xs:attribute name='dynamic' type='xs:boolean' default='false'/>
</xs:complexType>
<xs:simpleType name='handlingType'>
   <xs:restriction base='xs:string'>
      <xs:enumeration value='synchronous'/>
      <xs:enumeration value='asynchronous'/>
      <xs:enumeration value='delayed'/>
   </xs:restriction>
</xs:simpleType>
<xs:simpleType name='modeType'>
   <xs:restriction base='xs:string'>
```

```
      <xs:enumeration value='middle'/>
      <xs:enumeration value='activation'/>
      <xs:enumeration value='terminal'/>
   </xs:restriction>
</xs:simpleType>
<xs:simpleType name='directionType'>
   <xs:restriction base='xs:string'>
      <xs:enumeration value='in'/>
      <xs:enumeration value='out'/>
   </xs:restriction>
</xs:simpleType>
<xs:simpleType name='deliveryType'>
   <xs:restriction base='xs:string'>
      <xs:enumeration value='unreliable'/>
      <xs:enumeration value='reliable'/>
   </xs:restriction>
</xs:simpleType>
<xs:element name='exchange'>
   <xs:complexType>
      <xs:complexContent>
         <xs:extension base='documentation'>
            <xs:sequence>
               <xs:element name='paramSetRef' type='paramSetRefType' minOccurs='0'/>
               <xs:element name='cSetRefs' type='cSetRefsType' minOccurs='0'/>
               <xs:element name='endPointDescr' type='endPointDescrType' minOccurs='0'/>
               <xs:element name='replyToInform' type='endPointDescrType' minOccurs='0'/>
            </xs:sequence>
            <xs:attribute name='message' type='xs:QName' use='required'/>
            <xs:attribute name='direction' type='directionType' use='required'/>
            <xs:attribute name='mode' type='modeType' default='middle'/>
            <xs:attribute name='type' type='handlingType' default='asynchronous'/>
            <xs:attribute name='delivery' type='deliveryType' default='unreliable'/>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
</xs:element>
<xs:group name='actionGroup'>
   <xs:choice>
      <xs:element ref='exchange'/>
      <xs:element ref='raise'/>
      <xs:element ref='delayFor'/>
      <xs:element ref='delayUntil'/>
   </xs:choice>
</xs:group>
<xs:element name='delayFor'>
   <xs:complexType>
      <xs:complexContent>
         <xs:extension base='documentation'>
            <xs:sequence/>
            <xs:attribute name='timeout' type='xs:QName'/>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
</xs:element>
<xs:element name='delayUntil'>
   <xs:complexType>
      <xs:complexContent>
         <xs:extension base='documentation'>
            <xs:sequence/>
            <xs:attribute name='deadline' type='xs:QName'/>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
</xs:element>
<xs:element name='raise'>
   <xs:complexType>
      <xs:complexContent>
         <xs:extension base='documentation'>
            <xs:sequence>
               <xs:element name='paramSetInst' type='paramSetInstType' minOccurs='0'/>
```

```
            </xs:sequence>
            <xs:attribute name='signal' type='xs:QName' use='required'/>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
</xs:element>
<xs:group name='protocolControlGroup'>
   <xs:choice>
      <xs:element ref='empty'/>
      <xs:element ref='threadOfInteraction'/>
      <xs:element ref='threadOfInteractionRef'/>
      <xs:element ref='protocolRef'/>
      <xs:element ref='switch'/>
      <xs:element ref='all'/>
      <xs:element ref='pick'/>
      <xs:element ref='while'/>
      <xs:element ref='context'/>
      <xs:element ref='compensate'/>
      <xs:element ref='synchronize'/>
      <xs:group ref='actionGroup'/>
   </xs:choice>
</xs:group>
<xs:element name='threadOfInteraction'>
   <xs:complexType>
      <xs:complexContent>
         <xs:extension base='documentation'>
            <xs:sequence>
               <xs:element name='paramSetDecl' type='paramSetDeclType' minOccurs='0'/>
               <xs:choice minOccurs='0' maxOccurs='unbounded'>
                  <xs:group ref='actionGroup'/>
                  <xs:group ref='protocolControlGroup'/>
               </xs:choice>
            </xs:sequence>
            <xs:attribute name='threadName' type='xs:NCName' use='optional'/>
            <xs:attribute name='activation' type='xs:boolean' use='optional' default='false'/>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
</xs:element>
<xs:element name='threadOfInteractionRef'>
   <xs:complexType>
      <xs:complexContent>
         <xs:extension base='documentation'>
            <xs:sequence>
               <xs:element name='paramSetInst' type='paramSetInstType' minOccurs='0'/>
            </xs:sequence>
            <xs:attribute name='threadRef' type='xs:QName'/>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
</xs:element>
<xs:element name='synchronize'>
   <xs:complexType>
      <xs:sequence>
         <xs:element name='tofRef' type='xs:QName' minOccurs='2' maxOccurs='unbounded'/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
<xs:complexType name='protocolType'>
   <xs:complexContent>
      <xs:extension base='documentation'>
         <xs:sequence>
            <xs:element name='name' type='xs:QName'/>
            <xs:element name='body' type='protocolBodyType'/>
         </xs:sequence>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
<xs:complexType name='protocolBodyType'>
   <xs:complexContent>
```

```
      <xs:extension base='documentation'>
         <xs:sequence>
            <xs:element name='header' type='headerType'/>
            <xs:element name='orchestation'>
               <xs:complexType>
                  <xs:sequence>
                     <xs:choice maxOccurs='unbounded'>
                        <xs:element ref='threadOfInteraction'/>
                        <xs:element ref='context'/>
                     </xs:choice>
                  </xs:sequence>
               </xs:complexType>
            </xs:element>
         </xs:sequence>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
<xs:complexType name='headerType'>
   <xs:complexContent>
      <xs:extension base='documentation'>
         <xs:sequence>
            <xs:element name='cSetDecl' type='cSetDeclType'
                        minOccurs='0' maxOccurs='unbounded'/>
            <xs:element name='paramSetDecl' type='paramSetDeclType'
                        minOccurs='0' maxOccurs='unbounded'/>
         </xs:sequence>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
<xs:element name='all'>
   <xs:complexType>
      <xs:sequence>
         <xs:group ref='protocolControlGroup' minOccurs='0' maxOccurs='unbounded'/>
         <xs:element ref='synchronize' minOccurs='0'/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
<xs:element name='action'>
   <xs:complexType>
      <xs:group ref='protocolControlGroup'/>
   </xs:complexType>
</xs:element>
<xs:element name='catch'>
   <xs:complexType>
      <xs:complexContent>
         <xs:extension base='documentation'>
            <xs:sequence>
               <xs:element name='paramSetRef' type='paramSetRefType' minOccurs='0'/>
            </xs:sequence>
            <xs:attribute name='exception' type='xs:QName' use='required'/>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
</xs:element>
<xs:group name='eventGroup'>
   <xs:choice>
      <xs:element ref='delayFor'/>
      <xs:element ref='delayUntil'/>
      <xs:element ref='exchange'/>
      <xs:element ref='catch'/>
   </xs:choice>
</xs:group>
<xs:complexType name='eventHandlerType'>
   <xs:sequence>
      <xs:element name='event'>
         <xs:complexType>
            <xs:group ref='eventGroup'/>
            <xs:attribute name='id' type='xs:QName'/>
         </xs:complexType>
      </xs:element>
```

```
      <xs:element ref='action'/>
   </xs:sequence>
</xs:complexType>
<xs:element name='pick'>
   <xs:complexType>
      <xs:complexContent>
         <xs:extension base='documentation'>
            <xs:sequence>
               <xs:element name='paramSetRef' type='paramSetRefType' minOccurs='0'/>
               <xs:element name='eventHandler' type='eventHandlerType' maxOccurs='unbounded'/>
               <xs:element name='onTimes' minOccurs='0'>
                  <xs:complexType>
                     <xs:group ref='protocolControlGroup'/>
                  </xs:complexType>
               </xs:element>
            </xs:sequence>
            <xs:attribute name='times' type='xs:string' use='optional'/>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
</xs:element>
<xs:complexType name='conditionType'>
   <xs:sequence>
      <xs:element name='paramSetRef' type='paramSetRefType' minOccurs='0'/>
   </xs:sequence>
   <xs:attribute name='condition' type='xs:string' use='required'/>
</xs:complexType>
<xs:complexType name='branchType'>
   <xs:complexContent>
      <xs:extension base='documentation'>
         <xs:sequence>
            <xs:element name='case' type='conditionType'/>
            <xs:element ref='action'/>
         </xs:sequence>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
<xs:element name='switch'>
   <xs:complexType>
      <xs:complexContent>
         <xs:extension base='documentation'>
            <xs:sequence>
               <xs:element name='branch' type='branchType' maxOccurs='unbounded'/>
               <xs:element name='default' minOccurs='0'>
                  <xs:complexType>
                     <xs:sequence>
                        <xs:element ref='action'/>
                     </xs:sequence>
                  </xs:complexType>
               </xs:element>
            </xs:sequence>
            <xs:attribute name='multiChoice' type='xs:boolean' default='false'/>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
</xs:element>
<xs:element name='while'>
   <xs:complexType>
      <xs:complexContent>
         <xs:extension base='documentation'>
            <xs:sequence>
               <xs:element name='condition' type='conditionType'/>
               <xs:element ref='action'/>
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
</xs:element>
<xs:element name='exception'>
   <xs:complexType>
```

```
        <xs:sequence>
           <xs:element name='paramSetDecl' type='paramSetDeclType' minOccurs='0'/>
           <xs:element ref='pick'/>
           <xs:element name='finally' minOccurs='0'>
              <xs:complexType>
                 <xs:group ref='protocolControlGroup'/>
              </xs:complexType>
           </xs:element>
        </xs:sequence>
     </xs:complexType>
  </xs:element>
  <xs:element name='empty'>
     <xs:complexType>
        <xs:complexContent>
           <xs:restriction base='documentation'>
              <xs:sequence/>
           </xs:restriction>
        </xs:complexContent>
     </xs:complexType>
  </xs:element>
  <xs:element name='context'>
     <xs:complexType>
        <xs:complexContent>
           <xs:extension base='documentation'>
              <xs:sequence>
                 <xs:element name='paramSetDecl' type='paramSetDeclType' minOccurs='0'/>
                 <xs:choice minOccurs='0' maxOccurs='unbounded'>
                    <xs:group ref='actionGroup'/>
                    <xs:group ref='protocolControlGroup'/>
                    <xs:element name='protocolRef' type='protocolRefType'/>
                 </xs:choice>
                 <xs:element ref='exception' minOccurs='0'/>
                 <xs:element name='compensation' type='compensationType' minOccurs='0'/>
              </xs:sequence>
           </xs:extension>
        </xs:complexContent>
     </xs:complexType>
  </xs:element>
  <xs:element name='compensate'>
     <xs:complexType>
        <xs:complexContent>
           <xs:extension base='documentation'>
              <xs:sequence>
                 <xs:element name='paramSetRef' type='paramSetRefType' minOccurs='0'/>
              </xs:sequence>
              <xs:attribute name='transactionRef' type='xs:NCName' use='required'/>
           </xs:extension>
        </xs:complexContent>
     </xs:complexType>
  </xs:element>
  <xs:complexType name='compensationType'>
     <xs:complexContent>
        <xs:extension base='documentation'>
           <xs:sequence>
              <xs:element name='paramSetRef' type='paramSetRefType' minOccurs='0'/>
              <xs:group ref='protocolControlGroup'/>
           </xs:sequence>
           <xs:attribute name='name' type='xs:NCName' use='required'/>
        </xs:extension>
     </xs:complexContent>
  </xs:complexType>
  <xs:element name='protocol' type='protocolType'/>
</xs:schema>
```

Apéndice C

Especificaciones ACSL

Índice General

C.1. Protocolo [FIPA] IteratedContractNet 255
C.2. Protocolo para compensación de tareas 263
C.3. Protocolo de aprendizaje cooperativo de Sian 268

C.1. Protocolo [FIPA] IteratedContractNet

En la fase de asignación, el agente que actúa como cabeza del holón difunde una solicitud de propuestas mediante un acto hablado *cfp* a todos los miembros del holón. Estos últimos evaluarán la solicitud y enviarán sus propuestas (acto *propose*) en caso de que que estén dispuestos a realizar la tarea solicitada en las condiciones propuestas. En otro caso rechazarán la solicitud (acto *refuse*). El agente *cabeza* evaluará entonces las propuestas recibidas y decidirá aceptar una (o más) de éstas (*accept*) y rechazar las restantes (*refuse*), o bien iniciar una nueva fase de solicitud modificando la solicitud inicial en busca de mejores propuestas. En este último caso se reduce el número de participantes, por lo que no puede hablarse de un protocolo de difusión y sí de un protocolo multipunto. El protocolo puede finalizar también si en algún momento se rechazan todas las propuestas.

C.1.1. Especificación ACSL

```
<protocol ...>
   <name>IteratedContractNet</name>
   <body>
      <header>
         <paramSetDecl name='protocolParams'>
            <param>
               <name>t</name>
               <type>urn:TIC2001-3451:schemas:acsl:types:task</type>
```

```
      </param>
      <param>
         <name>timeout</name>
         <type>urn:TIC2001-3451:schemas:acsl:types:duration</type>
      </param>
      <param>
         <name>numberOfAgents</name>
         <type>urn:TIC2001-3451:schemas:acsl:argumentTypes:twoOrMore</type>
         ---<xsd:simpleType xmlns:xsd='http://www.w3.org/2001/XMLSchema'>
                     <xsd:restriction base='xsd:positiveInteger'>
                         <xs:minInclusive value='2'/>
                     </xsd:restriction>
                 </xsd:simpleType>---
      </param>
   </paramSetDecl>
</header>
<orchestation>
   <threadOfInteraction threadName='Init' activation='true'>
      <paramSetDecl name='initParams'>
         <param>
            <name>t1</name>
            <type>urn:TIC2001-3451:schemas:acsl:types:task</type>
            <activationValue>t</activationValue>
         </param>
         <param>
            <name>n</name>
            <type>urn:TIC2001-3451:schemas:acsl:argumentTypes:twoOrMore</type>
            <activationValue>numberOfAgents</activationValue>
         </param>
      </paramSetDecl>
      <context>
         <exchange message='cfp' direction='out' mode='activation'
                   delivery='unreliable' type='asynchronous'>
            <paramSetRef>
               <paramRef mode='match'>t1</paramRef>
            </paramSetRef>
            <endPointDescr>
               <broadcastRef>
                  <structureRef>holonicIfAgents</structureRef>
                  <structureType>group</structureType>
               </broadcastRef>
            </endPointDescr>
         </exchange>
         <threadOfInteractionRef threadRef='WaitOpinion'>
            <paramSetInst>
               <paramInst>
                  <ref>t1</ref>
                  <value>t1</value>
               </paramInst>
               <paramInst>
                  <ref>n</ref>
                  <value>n</value>
               </paramInst>
            </paramSetInst>
         </threadOfInteractionRef>
         <exception>
            <paramSetDecl name='exceptionParams'>
               <param>
                  <name>c</name>
                  <type>urn:TIC2001-3451:schemas:acsl:types:communicative-act</type>
               </param>
            </paramSetDecl>
            <pick>
               <eventHandler>
                  <event>
                     <catch exception='NotUnderstoodException'/>
                  </event>
                  <action>
                     <exchange message='Not-Understood' direction='out' mode='terminal'
                               delivery='unreliable' type='asynchronous'>
```

```
                        <paramSetRef>
                           <paramRef mode='match'>c</paramRef>
                        </paramSetRef>
                     </exchange>
                  </action>
               </eventHandler>
            </pick>
         </exception>
      </context>
   </threadOfInteraction>
   <threadOfInteraction threadName='WaitOpinion'>
      <paramSetDecl name='waitOpinionParams'>
         <param>
            <name>t1</name>
            <type>urn:TIC2001-3451:schemas:acsl:types:task</type>
         </param>
         <param>
            <name>p</name>
            <type>urn:TIC2001-3451:schemas:acsl:types:proposal</type>
         </param>
         <param>
            <name>n</name>
            <type>urn:TIC2001-3451:schemas:acsl:argumentTypes:positiveInteger</type>
         </param>
         <param>
            <name>pl</name>
            <type>urn:TIC2001-3451:schemas:acsl:argumentTypes:proposal-List</type>
         </param>
      </paramSetDecl>
      <pick times='sub(n,1)'>
         <paramSetRef>
            <paramRef mode='match'>n</paramRef>
         </paramSetRef>
         <eventHandler>
            <event id='refuse'>
               <exchange message='Refuse' direction='in'>
                  <paramSetRef>
                     <paramRef mode='match'>t1</paramRef>
                  </paramSetRef>
               </exchange>
            </event>
            <action>
               <empty/>
            </action>
         </eventHandler>
         <eventHandler>
            <event id='propose'>
               <exchange message='Propose' direction='in'>
                  <paramSetRef>
                     <paramRef mode='match'>p(t1)</paramRef>
                     <paramRef mode='adjust'>p(t1)::pl</paramRef>
                  </paramSetRef>
               </exchange>
            </event>
            <action>
               <empty/>
            </action>
         </eventHandler>
         <eventHandler>
            <event>
               <exchange message='Any' direction='in' mode='middle'>
                  <paramSetRef>
                     <paramRef mode='adjust'>c</paramRef>
                  </paramSetRef>
               </exchange>
            </event>
            <action>
               <raise signal='NotUnderstoodException'>
                  <paramSetInst>
                     <paramInst>
```

```
                        <ref>c</ref>
                        <value>c</value>
                     </paramInst>
                  </paramSetInst>
               </raise>
            </action>
         </eventHandler>
         <onTimes>
            <switch multiChoice='false'>
               <branch>
                  <case condition='not(empty(pl))'>
                     <paramSetRef>
                        <paramRef mode='match'>pl</paramRef>
                     </paramSetRef>
                  </case>
                  <action>
                     <threadOfInteractionRef threadRef='OpinionProposals'>
                        <paramSetInst>
                           <paramInst>
                              <ref>t1</ref>
                              <value>t1</value>
                           </paramInst>
                           <paramInst>
                              <ref>pl</ref>
                              <value>pl</value>
                           </paramInst>
                        </paramSetInst>
                     </threadOfInteractionRef>
                  </action>
               </branch>
               <default>
                  <action>
                     <empty/>
                  </action>
               </default>
            </switch>
         </onTimes>
      </pick>
   </threadOfInteraction>
   <threadOfInteraction threadName='OpinionProposals'>
      <paramSetDecl name='OpinionProposalsParams'>
         <param>
            <name>t1</name>
            <type>urn:TIC2001-3451:schemas:acsl:types:task</type>
         </param>
         <param>
            <name>p</name>
            <type>urn:TIC2001-3451:schemas:acsl:types:proposal</type>
         </param>
         <param>
            <name>pl</name>
            <type>urn:TIC2001-3451:schemas:acsl:argumentTypes:task-proposal-List</type>
         </param>
         <param>
            <name>apl</name>
            <type>urn:TIC2001-3451:schemas:acsl:argumentTypes:task-proposal-List</type>
         </param>
         <param>
            <name>fpl</name>
            <type>urn:TIC2001-3451:schemas:acsl:argumentTypes:task-proposal-List</type>
         </param>
      </paramSetDecl>
      <switch>
         <branch>
            <case condition='WantToIterate'>
               <paramSetRef>
                  <paramRef mode='match'>t1</paramRef>
                  <paramRef mode='match'>pl</paramRef>
                  <paramRef mode='adjust'>t2</paramRef>
               </paramSetRef>
```

```
      </case>
      <action>
         <threadOfInteraction>
            <while>
               <condition condition='ExistProposalInProposals'>
                  <paramSetRef>
                     <paramRef mode='adjust'>p(t1)::pl</paramRef>
                     <paramRef mode='match'>t1</paramRef>
                  </paramSetRef>
               </condition>
               <action>
                  <switch multiChoice='false'>
                     <branch>
                        <case condition='WantToCounterPropose'>
                           <paramSetRef>
                              <paramRef mode='match'>p</paramRef>
                              <paramRef mode='match'>t2</paramRef>
                              <paramRef mode='adjust'>inc(n)</paramRef>
                           </paramSetRef>
                        </case>
                        <action>
                           <exchange message='cfp' direction='out'
                                     delivery='unreliable' mode='middle'
                                     type='asynchronous'>
                              <paramSetRef>
                                 <paramRef mode='match'>p.id</paramRef>
                                 <paramRef mode='match'>t2</paramRef>
                              </paramSetRef>
                           </exchange>
                        </action>
                     </branch>
                     <branch>
                        <case condition='WantToRejectProposal'>
                           <paramSetRef>
                              <paramRef mode='match'>p</paramRef>
                           </paramSetRef>
                        </case>
                        <action>
                           <exchange message='Reject' direction='out'
                                     delivery='unreliable' mode='middle'
                                     type='asynchronous'>
                              <paramSetRef>
                                 <paramRef mode='match'>p.id</paramRef>
                                 <paramRef mode='match'>p</paramRef>
                              </paramSetRef>
                           </exchange>
                        </action>
                     </branch>
                  </switch>
               </action>
            </while>
            <threadOfInteractionRef threadRef='WaitOpinion'>
               <paramSetInst>
                  <paramInst>
                     <ref>t1</ref>
                     <value>t2</value>
                  </paramInst>
                  <paramInst>
                     <ref>n</ref>
                     <value>n</value>
                  </paramInst>
               </paramSetInst>
            </threadOfInteractionRef>
         </threadOfInteraction>
      </action>
   </branch>
   <branch>
      <case condition='WantToFinalize'>
         <paramSetRef>
            <paramRef mode='match'>t1</paramRef>
```

```
      <paramRef mode='match'>pl</paramRef>
   </paramSetRef>
</case>
<action>
   <threadOfInteraction threadName='AcceptProposals'>
      <while>
         <condition condition='existProposalInProposals'>
            <paramSetRef>
               <paramRef mode='adjust'>p(t1)::pl</paramRef>
               <paramRef mode='match'>t1</paramRef>
            </paramSetRef>
         </condition>
         <action>
            <switch multiChoice='false'>
               <branch>
                  <case condition='WantToAcceptProposal'>
                     <paramSetRef>
                        <paramRef mode='match'>p</paramRef>
                        <paramRef mode='adjust'>p---apl</paramRef>
                     </paramSetRef>
                  </case>
                  <action>
                     <exchange message='Accept' direction='out'
                               delivery='unreliable' mode='middle'
                               type='asynchronous'>
                        <paramSetRef>
                           <paramRef mode='match'>p.id</paramRef>
                           <paramRef mode='match'>p</paramRef>
                        </paramSetRef>
                     </exchange>
                  </action>
               </branch>
               <branch>
                  <case condition='WantToRejectProposal'>
                     <paramSetRef>
                        <paramRef mode='match'>p</paramRef>
                     </paramSetRef>
                  </case>
                  <action>
                     <exchange message='Reject' direction='out'
                               delivery='unreliable' mode='middle'
                               type='asynchronous'>
                        <paramSetRef>
                           <paramRef mode='match'>p.id</paramRef>
                           <paramRef mode='match'>p</paramRef>
                        </paramSetRef>
                     </exchange>
                  </action>
               </branch>
            </switch>
         </action>
      </while>
      <context>
         <pick times='length(apl)'>
            <paramSetRef>
               <paramRef mode='match'>apl</paramRef>
            </paramSetRef>
            <eventHandler>
               <event>
                  <exchange message='Inform' direction='in' mode='middle'>
                     <paramSetRef>
                        <paramRef mode='adjust'>p(t1)---ipl</paramRef>
                        <paramRef mode='match'>t1</paramRef>
                     </paramSetRef>
                  </exchange>
               </event>
               <action>
                  <empty/>
               </action>
            </eventHandler>
```

```
      <eventHandler>
         <event>
            <exchange message='Failure' direction='in' mode='middle'>
               <paramSetRef>
                  <paramRef mode='adjust'>p(t1)---fpl</paramRef>
                  <paramRef mode='match'>t1</paramRef>
               </paramSetRef>
            </exchange>
         </event>
         <action>
            <protocolRef ref='TaskNegotiation'>
               <paramSetInst>
                  <paramInst>
                     <ref>role1</ref>
                     <value>this</value>
                  </paramInst>
                  <paramInst>
                     <ref>role2</ref>
                     <value>id</value>
                  </paramInst>
                  <paramInst>
                     <ref>proposal</ref>
                     <value>p</value>
                  </paramInst>
               </paramSetInst>
            </protocolRef>
         </action>
      </eventHandler>
      <eventHandler>
         <event>
            <exchange message='Any' direction='in' mode='middle'>
               <paramSetRef>
                  <paramRef mode='adjust'>c</paramRef>
               </paramSetRef>
            </exchange>
         </event>
         <action>
            <raise signal='NotUnderstoodException'>
               <paramSetInst>
                  <paramInst>
                     <ref>c</ref>
                     <value>c</value>
                  </paramInst>
               </paramSetInst>
            </raise>
         </action>
      </eventHandler>
      <eventHandler>
         <event>
            <delayUntil deadline='timeout'/>
         </event>
         <action>
            <raise signal='UninformedTasks'>
               <paramSetInst>
                  <paramInst>
                     <ref>pl</ref>
                     <value>apl-ipl-fpl</value>
                  </paramInst>
               </paramSetInst>
            </raise>
         </action>
      </eventHandler>
   </pick>
   <exception>
      <paramSetDecl name='exceptionParams'>
         <param>
            <name>pl</name>
            <type>urn:TIC2001-3451:schemas:acsl:argumentTypes:
                                 task-proposal-List
            </type>
```

```
                        </param>
                     </paramSetDecl>
                     <pick>
                        <eventHandler>
                           <event>
                              <catch exception='UninformedTasks'>
                                 <paramSetRef>
                                    <paramRef mode='adjust'>pl</paramRef>
                                 </paramSetRef>
                              </catch>
                           </event>
                           <action>
                              <while>
                                 <condition condition='existProposalInProposals'>
                                    <paramSetRef>
                                       <paramRef mode='adjust'>p::pl</paramRef>
                                    </paramSetRef>
                                 </condition>
                                 <action>
                                    <compensate transactionRef='TaskCompensation'/>
                                 </action>
                              </while>
                           </action>
                        </eventHandler>
                     </pick>
                  </exception>
                  <compensation name='TaskCompensation'>
                     <protocolRef ref='TaskNegotiation'>
                        <paramSetInst>
                           <paramInst>
                              <ref>role1</ref>
                              <value>this</value>
                           </paramInst>
                           <paramInst>
                              <ref>role2</ref>
                              <value>id</value>
                           </paramInst>
                           <paramInst>
                              <ref>proposal</ref>
                              <value>p</value>
                           </paramInst>
                        </paramSetInst>
                     </protocolRef>
                  </compensation>
               </context>
            </threadOfInteraction>
         </action>
      </branch>
   </switch>
</threadOfInteraction>
</orchestation>
</body>
</protocol>
```

C.1.2. Especificación del intérprete SOE

$$\langle\langle \text{Init}, t_1 \in Tasks, N\rangle, WantProposals(t_1)\rangle \xrightarrow{cfp(t_1)} \langle\langle \text{WaitOpinion}, t_1, N, []\rangle, NOP\rangle$$

$$\langle\langle \text{WaitOpinion}, t_1, succ(N), l\rangle, true\rangle \xrightarrow[Refuse(t_1)]{} \langle\langle \text{WaitOpinion}, t_1, N, l\rangle, NOP\rangle$$

$$\langle\langle \text{WaitOpinion}, t_1, succ(N), l\rangle, true\rangle \xrightarrow[Propose(p(t_1))]{} \langle\langle \text{WaitOpinion}, t_1, N, p(t_1) :: l\rangle, NOP\rangle$$

$$\langle\langle \text{WaitOpinion}, t_1, zero, l\rangle, true\rangle \xrightarrow[Refuse(t_1)]{} \langle\langle \text{OpinionProposals}, t_1, l\rangle, NOP\rangle$$

$$\langle\langle \text{WaitOpinion}, t_1, zero, l\rangle, true\rangle \xrightarrow[Propose(p(t_1))]{} \langle\langle \text{OpinionProposals}, t_1, p(t_1) :: l\rangle, NOP\rangle$$

$$\langle\langle \text{WaitOpinion}, t_1, zero, []\rangle, true\rangle \xrightarrow[Refuse(t_1)]{} \langle\langle \text{End}, \rangle, NOP\rangle$$

$$\langle\langle \text{WaitOpinion}, t_1, zero, []\rangle, true\rangle \xrightarrow[Propose(p(t_1))]{} \langle\langle \text{OpinionProposals}, t_1, p(t_l) :: []\rangle, NOP\rangle$$

$\langle\langle \text{OpinionProposals}, t_1, l\rangle, WantToIterate(l, t_2)\rangle \overrightarrow{\tau} \langle\langle \text{IterateProposals}, t_2, N, []\rangle, \text{NOP}\rangle$

$\langle\langle \text{OpinionProposals}, t_1, l\rangle, \sim WantToIterate(l, t_2)\rangle \overrightarrow{\tau} \langle\langle \text{AcceptProposals}, t_1, l, 0\rangle, \text{NOP}\rangle$

$\langle\langle \text{AcceptProposals}, t_1, p(t_1) :: l, N\rangle, WTAcpProp(p(t_1))\rangle \xrightarrow{Accept(p(t_1))} \langle\langle \text{AcceptProposals}, t_1, l, Succ(N)\rangle, NOP\rangle$

$\langle\langle \text{AcceptProposals}, t_1, p(t_1) :: l, N\rangle, \sim WTRejProp(p(t_1))\rangle \xrightarrow{Reject(p(t_1))} \langle\langle \text{AcceptProposals}, t_1, l, Succ(N)\rangle, NOP\rangle$

$\langle\langle \text{AcceptProposals}, t_1, [], N\rangle, true\rangle \overrightarrow{\tau} \langle\langle \text{WaitCompletion}, N\rangle, \text{NOP}\rangle$

$\langle\langle \text{IterateProposals}, t_1, p(t_1) :: l, M\rangle, WTIterProp(p(t_1))\rangle \xrightarrow{cfp(t_1)} \langle\langle \text{IterateProposals}, t_1, l, succ(M)\rangle, NOP\rangle$

$\langle\langle \text{IterateProposals}, t_1, p(t_1) :: l, M\rangle, \sim WTIterProp(p(t_1))\rangle \xrightarrow{Reject(p(t_1))} \langle\langle \text{IterateProposals}, t_1, l, M\rangle, NOP\rangle$

$\langle\langle \text{IterateProposals}, t_1, [], M\rangle, true\rangle \overrightarrow{\tau} \langle\langle \text{WaitOpinion}, t_1, M, []\rangle, \text{NOP}\rangle$

$\langle\langle \text{WaitCompletion}, t_1, succ(N), apl, fpl\rangle, true\rangle \xrightarrow[Inform(p(t_1))]{} \langle\langle \text{WaitCompletion}, t_1, N, p(t_1) :: apl, fpl\rangle, NOP\rangle$

$\langle\langle \text{WaitCompletion}, t_1, succ(N), apl, fpl\rangle, true\rangle \xrightarrow[Failure(p(t_1))]{} \langle\langle \text{WaitCompletion}, t_1, N, apl, p(t_1) :: fpl\rangle, NOP\rangle$

$\langle\langle \text{WaitCompletion}, t_1, Zero, apl, p :: fpl\rangle, true\rangle \overrightarrow{\tau} \langle\langle \text{Compensation}, t_1, fpl\rangle, \text{NOP}\rangle$

$\langle\langle \text{WaitCompletion}, t_1, Zero, apl, []\rangle, true\rangle \overrightarrow{\tau} \langle\langle \text{End}, \rangle, \text{NOP}\rangle$

C.2. Protocolo para compensación de tareas

El protocolo comienza cuando un agente recibe de una propuesta para realizar dicha tarea bajo unas ciertas condiciones. El agente implicado debe entonces evaluar la propuesta y tomar una decisión, pudiendo como consecuencia aceptar la tarea, rechazarla (en cuyo caso finaliza el protocolo y, posiblemente, se inicia una nueva fase de asignación para dicha tarea) o realizar una contra-propuesta *counter-propose*) que satisfaga parcialmente la meta de la propuesta original. En caso de aceptar la propuesta, el agente tiene aún la libertad de desdecirse y cancelar su aceptación (nivel de deseos) mediante un acto *cancel* , o bien comprometerse a satisfacer la propuesta (nivel de intenciones) mediante un acto *commit*. Si el agente acuerda satisfacer la propuesta, se habrá producido una [nueva] asignación correcta. El agente deberá entonces informar acerca de la terminación de dicha asignación, comunicando si esta ha sido satisfecha (*inform* o *satisfy*) o ha fallado (*Failure*). También se contempla la posibilidad de que un agente cancele un compromiso contraído (*de-commit*). Esta cancelación conlleva diferentes consecuencias que la cancelación de una aceptación (*cancel*). Si se realiza una contra-propuesta, el agente iniciador deberá opinar acerca de la misma, bien aceptándola, bien rechazándola, en cuyo caso, tiene la opción de realizar una nueva contra-propuesta o terminar la fase de compensación y volver a la fase de asignación.

Nota. Como ejemplo de protocolo de compensación: Qué pasa con la p eliminada de la B_e tras una propuesta/solicitud de modificación si el agente que inicia el protocolo decide terminar éste sin que se haya alcanzado ningún consenso o en el caso de que se produzca una excepción.

C.2.1. Especificación ACSL

```
<protocol ...>
    <name>taskNegotiation</name>
    <body>
        <header>
            <paramSetDecl name='protocolParams'>
                <param>
                    <name>t</name>
                    <type>urn:TIC2001-3451:schemas:acsl:types:task</type>
                </param>
                <param>
                    <name>r</name>
                    <type>urn:TIC2001-3451:schemas:acsl:types:task</type>
                </param>
                <param>
                    <name>timeout</name>
                    <type>urn:TIC2001-3451:schemas:acsl:types:duration</type>
                </param>
            </paramSetDecl>
        </header>
        <orchestation>
            <threadOfInteraction threadName='Init'>
                <paramSetDecl name='initParams'>
                    <param>
                        <name>t</name>
                        <type>urn:TIC2001-3451:schemas:acsl:types:task</type>
                    </param>
                </paramSetDecl>
                <context>
                    <exchange message='Propose' direction='out' mode='activation'
                              delivery='unreliable' type='asynchronous'>
                        <paramSetRef>
                            <paramRef>t</paramRef>
                        </paramSetRef>
                    </exchange>
                    <threadOfInteractionRef threadRef='WaitOpinion'>
                        <paramSetInst>
                            <paramInst>
                                <ref>t</ref>
                                <value>t</value>
                            </paramInst>
                        </paramSetInst>
                    </threadOfInteractionRef>
                    <exception>
                        <pick>
                            <eventHandler>
                                <event>
                                    <catch exception='De-CommitException'/>
                                </event>
                                <action>
                                    <compensate transactionRef='compensateTask'/>
                                </action>
                            </eventHandler>
                        </pick>
                    </exception>
                    <compensation name='compensateTask'>
                        <empty/>
                    </compensation>
                </context>
            </threadOfInteraction>
            <threadOfInteraction threadName='WaitOpinion'>
                <paramSetDecl name='waitOpinionParams'>
                    <param>
                        <name>t</name>
                        <type>urn:TIC2001-3451:schemas:acsl:types:task</type>
                    </param>
                </paramSetDecl>
                <pick>
                    <eventHandler>
```

```
                <event>
                    <exchange message='Reject' direction='in' mode='middle'
                              delivery='unreliable' type='asynchronous'>
                        <paramSetRef>
                            <paramRef>t</paramRef>
                        </paramSetRef>
                    </exchange>
                </event>
                <action>
                    <empty/>
                </action>
            </eventHandler>
            <eventHandler>
                <event>
                    <exchange message='Accept' direction='in' mode='middle'
                              delivery='unreliable' type='asynchronous'>
                        <paramSetRef>
                            <paramRef>t</paramRef>
                        </paramSetRef>
                    </exchange>
                </event>
                <action>
                    <threadOfInteractionRef threadRef='WaitCommitment'>
                        <paramSetInst>
                            <paramInst>
                                <ref>t</ref>
                                <value>t</value>
                            </paramInst>
                        </paramSetInst>
                    </threadOfInteractionRef>
                </action>
            </eventHandler>
            <eventHandler>
                <event>
                    <exchange message='CounterPropose' direction='in' mode='middle'
                              delivery='unreliable' type='asynchronous'>
                        <paramSetRef>
                            <paramRef>t</paramRef>
                            <paramRef>r</paramRef>
                        </paramSetRef>
                    </exchange>
                </event>
                <action>
                    <threadOfInteractionRef threadRef='SendDecision'>
                        <paramSetInst>
                            <paramInst>
                                <ref>t</ref>
                                <value>r</value>
                            </paramInst>
                        </paramSetInst>
                    </threadOfInteractionRef>
                </action>
            </eventHandler>
        </pick>
    </threadOfInteraction>
    <threadOfInteraction threadName='WaitCommitment'>
        <paramSetDecl name='waitCommitmentParams'>
            <param>
                <name>t</name>
                <type>urn:TIC2001-3451:schemas:acsl:types:task</type>
            </param>
        </paramSetDecl>
        <pick>
            <eventHandler>
                <event>
                    <exchange message='Commit' direction='in' mode='middle'
                              delivery='unreliable' type='asynchronous'>
                        <paramSetRef>
                            <paramRef>t</paramRef>
                        </paramSetRef>
```

```
                </exchange>
            </event>
            <action>
                <threadOfInteractionRef threadRef='WaitResult'>
                    <paramSetInst>
                        <paramInst>
                            <ref>t</ref>
                            <value>t</value>
                        </paramInst>
                    </paramSetInst>
                </threadOfInteractionRef>
            </action>
        </eventHandler>
        <eventHandler>
            <event>
                <exchange message='Cancel' direction='in' mode='middle'
                          delivery='unreliable' type='asynchronous'>
                    <paramSetRef>
                        <paramRef>t</paramRef>
                    </paramSetRef>
                </exchange>
            </event>
            <action>
                <empty/>
            </action>
        </eventHandler>
    </pick>
</threadOfInteraction>
<threadOfInteraction threadName='WaitResult'>
    <paramSetDecl name='waitResultParams'>
        <param>
            <name>t</name>
            <type>urn:TIC2001-3451:schemas:acsl:types:task</type>
        </param>
        <param>
            <name>r</name>
            <type>urn:TIC2001-3451:schemas:acsl:types:task</type>
        </param>
    </paramSetDecl>
    <pick>
        <eventHandler>
            <event>
                <exchange message='Fail' direction='in' mode='middle'
                          delivery='unreliable' type='asynchronous'>
                    <paramSetRef>
                        <paramRef>t</paramRef>
                    </paramSetRef>
                </exchange>
            </event>
            <action>
                <empty/>
            </action>
        </eventHandler>
        <eventHandler>
            <event>
                <exchange message='Satisfy' direction='in' mode='middle'
                          delivery='unreliable' type='asynchronous'>
                    <paramSetRef>
                        <paramRef>t</paramRef>
                    </paramSetRef>
                </exchange>
            </event>
            <action>
                <empty/>
            </action>
        </eventHandler>
        <eventHandler>
            <event>
                <exchange message='De-Commit' direction='in' mode='middle'
                          delivery='unreliable' type='asynchronous'>
```

```
                    <paramSetRef>
                        <paramRef>t</paramRef>
                    </paramSetRef>
                </exchange>
            </event>
            <action>
                <raise signal='De-CommitException'/>
            </action>
        </eventHandler>
    </pick>
</threadOfInteraction>
<threadOfInteraction threadName='SendDecision'>
    <paramSetDecl name='sendDecisionParams'>
        <param>
            <name>t</name>
            <type>urn:TIC2001-3451:schemas:acsl:types:task</type>
        </param>
        <param>
            <name>r</name>
            <type>urn:TIC2001-3451:schemas:acsl:types:task</type>
        </param>
    </paramSetDecl>
    <switch multiChoice='false'>
        <branch>
            <case condition='WantToAccept'>
                <paramSetRef>
                    <paramRef>t</paramRef>
                </paramSetRef>
            </case>
            <action>
                <threadOfInteraction threadName='Accept'>
                    <exchange message='Accept' direction='out' mode='middle'
                              delivery='unreliable' type='asynchronous'>
                        <paramSetRef>
                            <paramRef>t</paramRef>
                        </paramSetRef>
                    </exchange>
                    <threadOfInteractionRef threadRef='WaitCommitment'>
                        <paramSetInst>
                            <paramInst>
                                <ref>t</ref>
                                <value>t</value>
                            </paramInst>
                        </paramSetInst>
                    </threadOfInteractionRef>
                </threadOfInteraction>
            </action>
        </branch>
        <branch>
            <case condition='WantToReject'>
                <paramSetRef>
                    <paramRef>t</paramRef>
                </paramSetRef>
            </case>
            <action>
                <exchange message='Reject' direction='out' mode='terminal'
                          delivery='unreliable' type='asynchronous'>
                    <paramSetRef>
                        <paramRef>t</paramRef>
                    </paramSetRef>
                </exchange>
            </action>
        </branch>
        <branch>
            <case condition='WantToCounterPropose'>
                <paramSetRef>
                    <paramRef>t</paramRef>
                </paramSetRef>
            </case>
            <action>
```

```
                        <threadOfInteraction threadName='CounterPropose'>
                            <exchange message='CounterPropose' direction='out' mode='middle'
                                      delivery='unreliable' type='asynchronous'>
                                <paramSetRef>
                                    <paramRef>t</paramRef>
                                    <paramRef>r</paramRef>
                                </paramSetRef>
                            </exchange>
                            <threadOfInteractionRef threadRef='WaitOpinion'>
                                <paramSetInst>
                                    <paramInst>
                                        <ref>t</ref>
                                        <value>r</value>
                                    </paramInst>
                                </paramSetInst>
                            </threadOfInteractionRef>
                        </threadOfInteraction>
                    </action>
                </branch>
            </switch>
        </threadOfInteraction>
    </orchestation>
</body>
</protocol>
```

C.2.2. Especificación del intérprete SOE

$\langle\langle \text{Init}, t_1 \in Tasks\rangle, WantToPropose(t_1)\rangle \xrightarrow{Propose(t_1)} \langle\langle \text{WaitOpinion}, t_1\rangle, NOP\rangle$

$\langle\langle \text{WaitOpinion}, t_1\rangle, true\rangle \xrightarrow[Reject(t_1)]{} \langle\langle \text{End}, t_1\rangle, NOP\rangle$

$\langle\langle \text{WaitOpinion}, t_1\rangle, true\rangle \xrightarrow[Accept(t_1)]{} \langle\langle \text{WaitCommitment}, t_1\rangle, NOP\rangle$

$\langle\langle \text{WaitOpinion}, t_1\rangle, true\rangle \xrightarrow[CounterPropose(t_1,t_2)]{} \langle\langle \text{SendDecision}, t_2\rangle, NOP\rangle$

$\langle\langle \text{WaitCommitment}, t_1\rangle, true\rangle \xrightarrow[Commit(t_1)]{} \langle\langle \text{WaitResult}, t_1\rangle, NOP\rangle$

$\langle\langle \text{WaitCommitment}, t_1\rangle, true\rangle \xrightarrow[Cancel(t_1)]{} \langle\langle \text{End}, t_1\rangle, NOP\rangle$

$\langle\langle \text{WaitResult}, t_1\rangle, true\rangle \xrightarrow[Fail(t_1)]{} \langle\langle \text{End}, t_1\rangle, NOP\rangle$

$\langle\langle \text{WaitResult}, t_1\rangle, true\rangle \xrightarrow[Satisfy(t_1)]{} \langle\langle \text{End}, t_1\rangle, t_1 \rightarrow (Done)\rangle$

$\langle\langle \text{WaitResult}, t_1\rangle, true\rangle \xrightarrow[De-Commit(t_1)]{} \langle\langle \text{End}, t_1\rangle, \text{Reputation} \rightarrow \text{id}\rangle$

$\langle\langle \text{SendDecision}, t_1\rangle, WantToAccpet(t_1)\rangle \xrightarrow{Accept(t_1)} \langle\langle \text{WaitCommitment}, t_1\rangle, t_1 \rightarrow \text{Tasks}\rangle$

$\langle\langle \text{SendDecision}, t_1\rangle, WantToCounterPropose(t_1, t_2)\rangle \xrightarrow{CounterPropose(t_1,t_2)} \langle\langle \text{WaitOpinion}, t_2\rangle, NOP\rangle$

C.3. Protocolo de aprendizaje cooperativo de Sian

Los protocolos de interacción estandarizados por FIPA se han mostrado adecuados como prueba de concepto para ilustrar el poder expresivo de la notación gráfica AUML. Sin embargo, adolecen en general del grado de complejidad necesario para mostrar el poder expresivo del lenguaje ACSL. A lo largo de los años se han desarrollado numerosos protocolos de cooperación que, pese a presentar un fuerte componente de interacción y haber sido comúnmente aceptados y utilizados por la comunidad internacional de DAI, no han sido incluidos hasta la fecha en la biblioteca de protocolos de interacción de FIPA. El protocolo de aprendizaje cooperativo

desarrollado por Sati Singh Sian [Sia91] constituye un ejemplo paradigmático, por lo que se ha escogido para ilustrar la utilización de ACSL como lenguaje de especificación. Se trata de un protocolo de aprendizaje en el que dos o más agentes dialogan con el fin de alcanzar acuerdos acerca de las diferentes hipótesis elaboradas por cada uno de ellos. A la hora de describir este protocolo conviene diferenciar dos posibles escenarios de aplicación que derivarán en dos especificaciones diferentes: el protocolo seguido por dos agentes y el protocolo seguido por n agentes (para $n > 2$).

C.3.1. Descripción de las acciones comunicativas utilizadas

En la descripción del protocolo de coordinación de Singh Sian, se asume la existencia de un conjunto de actos hablados, de mayor nivel que los disponibles en KQML o FIPA-ACL, que definen [un subconjunto de] las acciones comunicativas disponibles en cada uno de los agentes que ofrece este protocolo como interfaz de comunicación. A continuación se describe de manera informal la semántica de cada uno de estos actos hablados (representados como performativas del ACL correspondiente):

C.3.2. Descripción del protocolo para dos agentes

Cada uno de los agentes dispone de una base de experiencia B_e y de una base de conocimiento B_c. En el momento en que uno de los dos agentes desee proponer una hipótesis elaborada a partir del conocimiento acumulado en su B_e, iniciará el protocolo enviando al otro agente un mensaje con su propuesta p. El otro agente debe entonces contestar confirmando dicha hipótesis si está de acuerdo con su verdad en función del conocimiento acumulado en su B_e o en su B_c privadas, mostrando su desacuerdo si dispone de una hipótesis opuesta en su B_c, proponiendo una modificación si es capaz de elaborar una propuesta más general o más específica a partir de su B_c, o indicando su incapacidad para opinar acerca de dicha hipótesis en cualquier otro caso. A partir de la opinión vertida por este último, el agente que formuló la hipótesis inicial evaluará su verdad en función de un determinado criterio preestablecido, pudiendo abandonarla (eliminándola de su B_e) o confirmarla (pasándola a su B_c). La decisión tomada debe ser comunicada al otro agente para que éste, que en caso de confirmación deberá comunicar su aceptación. En caso de que la opinión vertida por el segundo agente sea una propuesta de modificación de la hipótesis inicial por otra nueva hipótesis, se iniciará un proceso isomórfico al presentado en el que el agente que realizó la propuesta inicial deberá opinar acerca de la nueva hipótesis y será el otro agente quien deba evaluar su verdad a partir de la opinión vertida por el primero. Este proceso se repetirá hasta que alguno de los dos agentes

cese en sus propuestas de modificación y decida confirmar, oponerse o abstenerse con respecto a la hipótesis en curso.

Al margen de este proceso de aprendizaje cooperativo, cualquiera de los dos agentes podrá aseverar una proposición p a partir del conocimiento acumulado en su B_c, en cuyo caso el otro agente simplemente comunicará su aceptación y la incluirá en su B_c (eliminándola de su B_e si fuese necesario).

C.3.3. Descripción del protocolo para n agentes

El protocolo se inicia cuando uno de los agentes A_i participantes desea recabar la opinión del resto acerca de una hipótesis propia p elaborada a partir del conocimiento recopilado en su base de experiencia B_e. El agente iniciador difunde entonces una proposición al resto de agentes que, una vez evaluada, contestarán confirmado dicha hipótesis, mostrando su desacuerdo, señalando su falta de opinión o proponiendo su modificación por otra proposición q más general o más específica, siempre en función del contenido de sus bases de conocimiento. A_i esperará entonces por todas estas respuestas durante un tiempo limitado. En caso de no recibir ninguna petición de modificación, A_i deberá tomar una decisión acerca de la verdad de p y notificársela al resto de agentes confirmándo o retirando la proposición inicial. En caso de recibir alguna petición de modificación, el agente deberá escoger una en base a un criterio preestablecido (que en todo caso no forma parte del protocolo de coordinación) hacerla pública (identificando tanto la propuesta de modificación como el agente que la elaboró) y opinar acerca suyo confirmándola, mostrando su desacuerdo, su falta de opinión o proponiendo una nueva modificación.

El protocolo se complica ante la necesidad de contemplar la situación en que el agente recibe una indicación de una modificación, ya que esta indicación puede corresponder tanto a una solicitud de modificación elaborada por él mismo; en cuyo caso recabará la opinión del resto de agentes, como a una solicitud de modificación elaborada por otro agente; en cuyo caso le enviará su punto de vista.

La figura C.1 muestra la representación AUML del protocolo descrito.

Respecto de la representación AUML del protocolo de Sian para dos agentes, puede apreciarse que en esta ocasión el protocolo es simétrico para todos los agentes, por lo que todos ellos utilizarán la misma especificación ACSL.

C.3.4. Especificación ACSL

```
<protocol ...>
   <name>sianN</name>
   <body>
      <header>
         <paramSetDecl name="protocolParams">
            <param>
```

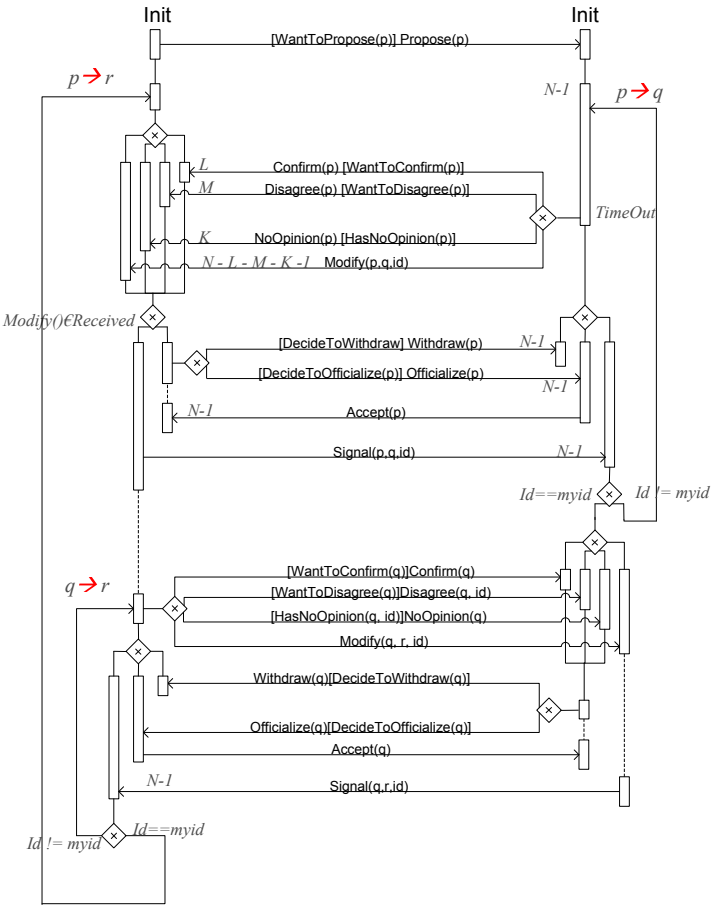

Figura C.1: Especificación AUML del protocolo de coordinación de Sian para más de dos agentes

```
            <name>p</name>
            <type>urn:TIC2001-3451:schemas:acsl:argumentTypes:proposition</type>
         </param>
         <param>
            <name>q</name>
            <type>urn:TIC2001-3451:schemas:acsl:argumentTypes:proposition</type>
         </param>
         <param>
            <name>timeout</name>
            <type>urn:TIC2001-3451:schemas:acsl:argumentTypes:duration</type>
         </param>
         <param>
            <name>numberOfAgents</name>
            <type>urn:TIC2001-3451:schemas:acsl:argumentTypes:twoOrMore</type>
         </param>
      </paramSetDecl>
   </header>
   <orchestation>
      <threadOfInteraction threadName="Init">
         <paramSetDecl name="initParams">
            <param>
               <name>p</name>
               <type>urn:TIC2001-3451:schemas:acsl:argumentTypes:proposition</type>
            </param>
         </paramSetDecl>
         <exchange message="Propose" direction="out" mode="activation"
                   delivery="unreliable" type="asynchronous">
```

```
            <paramSetRef>
               <paramRef>p</paramRef>
            </paramSetRef>
            <endPointDescr>
               <broadcastRef>
                  <structureRef>agents</structureRef>
                  <structureType>group</structureType>
               </broadcastRef>
            </endPointDescr>
         </exchange>
         <threadOfInteractionRef threadRef="OpinionPropose">
            <paramSetInst>
               <paramInst>
                  <ref>p</ref>
                  <value>p</value>
               </paramInst>
            </paramSetInst>
         </threadOfInteractionRef>
      </threadOfInteraction>
      <threadOfInteraction threadName="OpinionPropose">
         <paramSetDecl name="opinionProposeParams">
            <param>
               <name>p</name>
               <type>urn:TIC2001-3451:schemas:acsl:argumentTypes:proposition</type>
            </param>
            <param>
               <name>q</name>
               <type>urn:TIC2001-3451:schemas:acsl:argumentTypes:proposition</type>
            </param>
            <param>
               <name>ql</name>
               <type>urn:TIC2001-3451:schemas:acsl:argumentTypes:propositionList</type>
            </param>
         </paramSetDecl>
         <pick times="sub_numberOfAgents_1">
            <eventHandler>
               <event>
                  <exchange message="Confirm" direction="in" mode="middle">
                     <paramSetRef>
                        <paramRef>p</paramRef>
                     </paramSetRef>
                  </exchange>
               </event>
               <action>
                  <empty/>
               </action>
            </eventHandler>
            <eventHandler>
               <event>
                  <exchange message="Disagree" direction="in" mode="middle">
                     <paramSetRef>
                        <paramRef>p</paramRef>
                     </paramSetRef>
                  </exchange>
               </event>
               <action>
                  <empty/>
               </action>
            </eventHandler>
            <eventHandler>
               <event>
                  <exchange message="NoOpinion" direction="in" mode="middle">
                     <paramSetRef>
                        <paramRef>p</paramRef>
                     </paramSetRef>
                  </exchange>
               </event>
               <action>
                  <empty/>
               </action>
```

```
         </eventHandler>
         <eventHandler>
            <event id="modify">
               <exchange message="Modify" direction="in" mode="middle">
                  <paramSetRef>
                     <paramRef>p</paramRef>
                     <paramRef>lq</paramRef>
                  </paramSetRef>
               </exchange>
            </event>
            <action>
               <empty/>
            </action>
         </eventHandler>
         <eventHandler>
            <event>
               <delayFor timeout="timeout"/>
            </event>
            <action>
               <raise signal="timeoutReachedWaitingForOpinion"/>
            </action>
         </eventHandler>
         <onTimes>
            <switch multiChoice="false">
               <branch>
                  <case condition="greater_times_Modify_0"/>
                  <action>
                     <threadOfInteractionRef threadRef="OpinionAnswer">
                        <paramSetInst>
                           <paramInst>
                              <ref>p</ref>
                              <value>q_in_ql</value>
                           </paramInst>
                        </paramSetInst>
                     </threadOfInteractionRef>
                  </action>
               </branch>
               <default>
                  <action>
                     <threadOfInteractionRef threadRef="DecisionPropose">
                        <paramSetInst>
                           <paramInst>
                              <ref>p</ref>
                              <value>p</value>
                           </paramInst>
                        </paramSetInst>
                     </threadOfInteractionRef>
                  </action>
               </default>
            </switch>
         </onTimes>
      </pick>
   </threadOfInteraction>
   <threadOfInteraction threadName="DecisionPropose">
      <paramSetDecl name="decisionProposeParams">
         <param>
            <name>p</name>
            <type>urn:TIC2001-3451:schemas:acsl:parameterTypes:proposition</type>
         </param>
      </paramSetDecl>
      <switch multiChoice="false">
         <branch>
            <case condition="DecideToWithdraw">
               <paramSetRef>
                  <paramRef>p</paramRef>
               </paramSetRef>
            </case>
            <action>
               <exchange message="Withdraw" direction="out" delivery="unreliable"
                         mode="terminal" type="asynchronous">
```

```
                  <paramSetRef>
                     <paramRef>p</paramRef>
                  </paramSetRef>
                  <endPointDescr>
                     <broadcastRef>
                        <structureRef>agents</structureRef>
                        <structureType>group</structureType>
                     </broadcastRef>
                  </endPointDescr>
               </exchange>
            </action>
         </branch>
         <branch>
            <case condition="DecideToOfficialize">
               <paramSetRef>
                  <paramRef>p</paramRef>
               </paramSetRef>
            </case>
            <action>
               <threadOfInteraction>
                  <exchange message="Officialize" direction="out" delivery="unreliable"
                            mode="middle" type="asynchronous">
                     <paramSetRef>
                        <paramRef>p</paramRef>
                     </paramSetRef>
                     <endPointDescr>
                        <broadcastRef>
                           <structureRef>agents</structureRef>
                           <structureType>group</structureType>
                        </broadcastRef>
                     </endPointDescr>
                  </exchange>
                  <threadOfInteractionRef threadRef="AgreementPropose">
                     <paramSetInst>
                        <paramInst>
                           <ref>p</ref>
                           <value>q_in_ql</value>
                        </paramInst>
                     </paramSetInst>
                  </threadOfInteractionRef>
               </threadOfInteraction>
            </action>
         </branch>
      </switch>
   </threadOfInteraction>
   <threadOfInteraction threadName="OpinionAnswer">
      <paramSetDecl name="decisionProposeParams">
         <param>
            <name>p</name>
            <type>urn:TIC2001-3451:schemas:acsl:parameterTypes:proposition</type>
         </param>
         <param>
            <name>q</name>
            <type>urn:TIC2001-3451:schemas:acsl:parameterTypes:proposition</type>
         </param>
         <param>
            <name>id</name>
            <type>urn:TIC2001-3451:schemas:acsl:parameterTypes:agentID</type>
         </param>
         <param>
            <name>msg</name>
            <type>urn:TIC2001-3451:schemas:acsl:parameterTypes:message</type>
         </param>
      </paramSetDecl>
      <exchange message="signal" direction="out" delivery="unreliable"
                mode="middle" type="asynchronous">
         <paramSetRef>
            <paramRef>p</paramRef>
            <paramRef>lq</paramRef>
            <paramRef>id</paramRef>
```

```
      </paramSetRef>
      <endPointDescr>
         <broadcastRef>
            <structureRef>agents</structureRef>
            <structureType>group</structureType>
         </broadcastRef>
      </endPointDescr>
   </exchange>
   <switch multiChoice="false">
      <branch>
         <case condition="WantToConfirm">
            <paramSetRef>
               <paramRef>p</paramRef>
               <paramRef>msg</paramRef>
            </paramSetRef>
         </case>
         <action>
            <exchange message="Confirm" direction="out" delivery="unreliable"
                      mode="middle" type="asynchronous">
               <paramSetRef>
                  <paramRef>p</paramRef>
                  <paramRef>msg</paramRef>
                  --- Coge el valor de msg ---
               </paramSetRef>
               <endPointDescr>
                  <dynamicEndPointRef>
                     <rolePlayed>modifier</rolePlayed>
                     <endPointRef endPoint="reply-to">
                        <name>reply-to</name>
                        <XPathExpr>msg/header/reply-to</XPathExpr>
                     </endPointRef>
                  </dynamicEndPointRef>
               </endPointDescr>
            </exchange>
         </action>
      </branch>
      <branch>
         <case condition="WantToDisagree">
            <paramSetRef>
               <paramRef>p</paramRef>
               <paramRef>msg</paramRef>
            </paramSetRef>
         </case>
         <action>
            <exchange message="Disagree" direction="out" delivery="unreliable"
                      mode="middle" type="asynchronous">
               <paramSetRef>
                  <paramRef>p</paramRef>
                  <paramRef>msg</paramRef>
               </paramSetRef>
               <endPointDescr>
                  <dynamicEndPointRef>
                     <rolePlayed>modifier</rolePlayed>
                     <endPointRef endPoint="reply-to">
                        <name>reply-to</name>
                        <XPathExpr>msg/header/reply-to</XPathExpr>
                     </endPointRef>
                  </dynamicEndPointRef>
               </endPointDescr>
            </exchange>
         </action>
      </branch>
      <branch>
         <case condition="HaveNoOpinion">
            <paramSetRef>
               <paramRef>p</paramRef>
               <paramRef>msg</paramRef>
            </paramSetRef>
         </case>
         <action>
```

```
            <exchange message="NoOpinion" direction="out" delivery="unreliable"
                      mode="middle" type="asynchronous">
               <paramSetRef>
                  <paramRef>p</paramRef>
                  <paramRef>msg</paramRef>
                  --- Coge el valor de msg ---
               </paramSetRef>
               <endPointDescr>
                  <dynamicEndPointRef>
                     <rolePlayed>modifier</rolePlayed>
                     <endPointRef endPoint="reply-to">
                        <name>reply-to</name>
                        <XPathExpr>msg/header/reply-to</XPathExpr>
                     </endPointRef>
                  </dynamicEndPointRef>
               </endPointDescr>
            </exchange>
         </action>
      </branch>
      <branch>
         <case condition="WantToModify">
            <paramSetRef>
               <paramRef>p</paramRef>
               <paramRef>msg</paramRef>
            </paramSetRef>
         </case>
         <action>
            <exchange message="Modify" direction="out" delivery="unreliable"
                      mode="middle" type="asynchronous">
               <paramSetRef>
                  <paramRef>p</paramRef>
                  <paramRef>q</paramRef>
                  <paramRef>msg</paramRef>
               </paramSetRef>
               <endPointDescr>
                  <dynamicEndPointRef>
                     <rolePlayed>modifier</rolePlayed>
                     <endPointRef endPoint="reply-to">
                        <name>reply-to</name>
                        <XPathExpr>msg/header/reply-to</XPathExpr>
                     </endPointRef>
                  </dynamicEndPointRef>
               </endPointDescr>
            </exchange>
         </action>
      </branch>
   </switch>
   <threadOfInteractionRef threadRef="WaitAnswer">
      <paramSetInst>
         <paramInst>
            <ref>p</ref>
            <value>q</value>
         </paramInst>
      </paramSetInst>
   </threadOfInteractionRef>
</threadOfInteraction>
<threadOfInteraction threadName="AgreementPropose">
   <paramSetDecl name="decisionProposeParams">
      <param>
         <name>p</name>
         <type>urn:TIC2001-3451:schemas:acsl:parameterTypes:proposition</type>
      </param>
   </paramSetDecl>
   <pick times="sub(numberOfAgents,1)">
      <eventHandler>
         <event>
            <exchange message="Accept" direction="in" mode="terminal">
               <paramSetRef>
                  <paramRef>p</paramRef>
               </paramSetRef>
```

```
            </exchange>
         </event>
         <action>
            <empty/>
         </action>
      </eventHandler>
      <eventHandler>
         <event>
            <delayFor timeout="timeout"/>
         </event>
         <action>
            <raise signal="timeoutReachedWaitingForAccept"/>
         </action>
      </eventHandler>
      <onTimes>
         <empty/>
      </onTimes>
   </pick>
</threadOfInteraction>
<threadOfInteraction threadName="WaitAnswer">
   <paramSetDecl name="decisionProposeParams">
      <param>
         <name>p</name>
         <type>urn:TIC2001-3451:schemas:acsl:parameterTypes:proposition</type>
      </param>
      <param>
         <name>q</name>
         <type>urn:TIC2001-3451:schemas:acsl:parameterTypes:proposition</type>
      </param>
      <param>
         <name>id</name>
         <type>urn:TIC2001-3451:schemas:acsl:parameterTypes:agentID</type>
      </param>
      <param>
         <name>id</name>
         <type>urn:TIC2001-3451:schemas:acsl:parameterTypes:agentID</type>
      </param>
   </paramSetDecl>
   <pick>
      <eventHandler>
         <event>
            <exchange message="Withdraw" direction="in" mode="terminal">
               <paramSetRef>
                  <paramRef>p</paramRef>
               </paramSetRef>
            </exchange>
         </event>
         <action>
            <empty/>
         </action>
      </eventHandler>
      <eventHandler>
         <event>
            <exchange message="Officialize" direction="in" mode="middle">
               <paramSetRef>
                  <paramRef>p</paramRef>
               </paramSetRef>
            </exchange>
         </event>
         <action>
            <exchange message="Accept" direction="out" delivery="unreliable"
                      mode="terminal" type="asynchronous">
               <paramSetRef>
                  <paramRef>p</paramRef>
               </paramSetRef>
            </exchange>
         </action>
      </eventHandler>
      <eventHandler>
         <event>
```

```
                  <exchange message="Signal" direction="in" mode="middle">
                     <paramSetRef>
                        <paramRef>p</paramRef>
                        <paramRef>q</paramRef>
                        <paramRef>newid</paramRef>
                     </paramSetRef>
                  </exchange>
               </event>
               <action>
                  <switch multiChoice="false">
                     <branch>
                        <case condition="equals(id,newid)">
                           <paramSetRef>
                              <paramRef>id</paramRef>
                              <paramRef>newid</paramRef>
                           </paramSetRef>
                        </case>
                        <action>
                           <threadOfInteractionRef threadRef="OpinionPropose">
                              <paramSetInst>
                                 <paramInst>
                                    <ref>p</ref>
                                    <value>q</value>
                                 </paramInst>
                              </paramSetInst>
                           </threadOfInteractionRef>
                        </action>
                     </branch>
                     <default>
                        <action>
                           <threadOfInteractionRef threadRef="OpinionAnswer">
                              <paramSetInst>
                                 <paramInst>
                                    <ref>p</ref>
                                    <value>q</value>
                                 </paramInst>
                              </paramSetInst>
                           </threadOfInteractionRef>
                        </action>
                     </default>
                  </switch>
               </action>
            </eventHandler>
         </pick>
      </threadOfInteraction>
   </orchestation>
</body>
</protocol>
```

C.3.5. Especificación del intérprete SOE

Reglas del intérprete correspondiente al agente que inicia la conversación:

$\langle\langle \text{Init}, p_1 \in B_e\rangle, WantToPropose(p_1)\rangle \xrightarrow{Propose(p_1)} \langle\langle \text{Opinion}, p_1\rangle, NOP\rangle$

$\langle\langle \text{Init}, p_1 \in B_e\rangle, WantToAssert(p_1)\rangle \xrightarrow{Assert(p_1)} \langle\langle \text{AgreementPropose}, p_1\rangle, NOP\rangle$

$\langle\langle \text{AgreementPropose}, p_1\rangle, true\rangle \xrightarrow[Accept(p_1)]{} \langle\langle \text{End}, p_1\rangle, B_e - p_1, B_c + p_1\rangle$

$\langle\langle \text{AgreementPropose}, p_1\rangle, timeout\rangle \overrightarrow{\varepsilon} \langle\langle \text{End}, p_1\rangle, \text{NOP}\rangle$

$\langle\langle \text{Opinion}, p_1\rangle, true\rangle \xrightarrow[p_1, Confirm(p_1)]{} \langle\langle \text{DecisionPropose}, Confirm(p_1)\rangle, NOP\rangle$

$\langle\langle \text{Opinion}, p_1\rangle, true\rangle \xrightarrow[p_1, Disagree(p_1)]{} \langle\langle \text{DecisionPropose}, Disagree(p_1)\rangle, NOP\rangle$

$\langle\langle \text{Opinion}, p_1\rangle, true\rangle \xrightarrow[p_1, NoOpinion(p_1)]{} \langle\langle \text{DecisionPropose}, NoOpinion(p_1)\rangle, NOP\rangle$

$\langle\langle \text{Opinion}, p_1\rangle, true\rangle \xrightarrow[Modify(p_1, p_2)]{} \langle\langle \text{CounterOpinion}, p_2\rangle, B_e - p_1, B_c\rangle$

$\langle\langle \text{Opinion}, p_1\rangle, timeout\rangle \overrightarrow{\varepsilon} \langle\langle \text{End}, p_1\rangle, \text{NOP}\rangle$

$\langle\langle \text{DecisionPropose}, p_1, s(p_1)\rangle, DecideToWithdraw(p_1, s(p_1))\rangle \xrightarrow{Withdraw(p_1)} \langle\langle \text{End}, p_1\rangle, B_e - p_1, B_c\rangle$

$\langle\langle \text{DecisionPropose}, p_1, s(p_1)\rangle, DecideToOffic(p_1, s(p_1))\rangle \xrightarrow{Offic(p_1)} \langle\langle \text{AgreementPropose}, p_1\rangle, NOP\rangle$

$\langle\langle \text{DecisionAnswer}, p_1\rangle, true\rangle \xrightarrow[Withdraw(p_1)]{} \langle\langle \text{End}, p_1\rangle, B_e - p_1, B_c\rangle$

$\langle\langle \text{DecisionAnswer}, p_1\rangle, true\rangle \xrightarrow[Offic(p_1)]{} \langle\langle \text{AgreementAnswer}, p_1\rangle, NOP\rangle$

$\langle\langle \text{DecisionAnswer}, p_1\rangle, timeout\rangle \overrightarrow{\varepsilon} \langle\langle \text{End}, p_1\rangle, \text{NOP}\rangle$

$\langle\langle \text{AgreementAnswer}, p_1\rangle, true\rangle \xrightarrow{Accept(p_1)} \langle\langle \text{End}, p_1\rangle, B_e - p_1, B_c + p_1\rangle$

$\langle\langle \text{CounterOpinion}, p_1\rangle, WantToConfirm(p_1)\rangle \xrightarrow{Confirm(p_1)} \langle\langle \text{DecisionAnswer}, p_1\rangle, NOP\rangle$

$\langle\langle \text{CounterOpinion}, p_1\rangle, WantToDisagree(p_1)\rangle \xrightarrow{Disagree(p_1)} \langle\langle \text{DecisionAnswer}, p_1\rangle, NOP\rangle$

$\langle\langle \text{CounterOpinion}, p_1\rangle, NoOpinion(p_1)\rangle \xrightarrow{NoOpinion(p_1)} \langle\langle \text{DecisionAnswer}, p_1\rangle, NOP\rangle$

$\langle\langle \text{CounterOpinion}, p_1\rangle, WantToModify(p_1, p_2)\rangle \xrightarrow{Modify(p_1,p_2)} \langle\langle \text{Opinion}, p_2\rangle, B_e - p, B_c\rangle$

Reglas del intérprete para el agente que recibe la propuesta inicial:

$\langle\langle \text{Init}, p_1 \in B_e\rangle, true\rangle \xrightarrow[Propose(p_1)]{} \langle\langle \text{Opinion}, p_1\rangle, NOP\rangle$

$\langle\langle \text{Init}, p_1 \in B_e\rangle, true\rangle \xrightarrow[Assert(p_1)]{} \langle\langle \text{AgreementPropose}, p_1\rangle, NOP\rangle$

$\langle\langle \text{AgreementPropose}, p_1\rangle, true\rangle \xrightarrow{Accept(p_1)} \langle\langle \text{End}, p_1\rangle, B_e - p_1, B_c + p_1\rangle$

$\langle\langle \text{Opinion}, p_1\rangle, WantToConfirm(p_1)\rangle \xrightarrow{Confirm(p_1)} \langle\langle \text{DecisionPropose}, p_1\rangle, NOP\rangle$

$\langle\langle \text{Opinion}, p_1\rangle, WantToDisagree(p_1)\rangle \xrightarrow{Disagree(p_1)} \langle\langle \text{DecisionPropose}, p_1\rangle, NOP\rangle$

$\langle\langle \text{Opinion}, p_1\rangle, HaveNoOpinion(p_1)\rangle \xrightarrow{NoOpinion(p_1)} \langle\langle \text{DecisionPropose}, p_1\rangle, NOP\rangle$

$\langle\langle \text{Opinion}, p_1\rangle, WantToModify(p_1, p_2)\rangle \xrightarrow{Modify(p_1,p_2)} \langle\langle \text{CounterOpinion}, p_2\rangle, B_e - p_1, B_c\rangle$

$\langle\langle \text{CounterOpinion}, p_1\rangle, true\rangle \xrightarrow[Confirm(p_1)]{} \langle\langle \text{DecisionAnswer}, Confirm(p_1)\rangle, NOP\rangle$

$\langle\langle \text{CounterOpinion}, p_1\rangle, true\rangle \xrightarrow[Disagree(p_1)]{} \langle\langle \text{DecisionAnswer}, Disagree(p_1)\rangle, NOP\rangle$

$\langle\langle \text{CounterOpinion}, p_1\rangle, true\rangle \xrightarrow[NoOpinion(p_1)]{} \langle\langle \text{DecisionAnswer}, NoOpinion(p_1)\rangle, NOP\rangle$

$\langle\langle \text{CounterOpinion}, p_1\rangle, true\rangle \xrightarrow[Modify(p_1,p_2)]{} \langle\langle \text{Opinion}, p_2\rangle, B_e - p, B_c\rangle$

$\langle\langle \text{CounterOpinion}, p_1\rangle, timeout\rangle \overrightarrow{\varepsilon} \langle\langle \text{End}, p_1\rangle, \text{NOP}\rangle$

$\langle\langle \text{DecisionPropose}, p_1\rangle, true\rangle \xrightarrow[Withdraw(p_1)]{} \langle\langle \text{End}, p_1\rangle, B_e - p_1, B_c\rangle$

$\langle\langle \text{DecisionPropose}, p_1\rangle, true\rangle \xrightarrow[Offic(p_1)]{} \langle\langle \text{AgreementPropose}, p_1\rangle, NOP\rangle$

$\langle\langle \text{DecisionPropose}, p_1\rangle, timeout\rangle \overrightarrow{\varepsilon} \langle\langle \text{End}, p_1\rangle, \text{NOP}\rangle$

$\langle\langle \text{AgreementAnswer}, p_1\rangle, true\rangle \xrightarrow[Accept(p_1)]{} \langle\langle \text{End}, p_1\rangle, B_e - p_1, B_c + p_1\rangle$

$\langle\langle \text{AgreementAnswer}, p_1\rangle, timeout\rangle \overrightarrow{\varepsilon} \langle\langle \text{End}, p_1\rangle, \text{NOP}\rangle$

$\langle\langle \text{DecisionAnswer}, p_1, s(p_1)\rangle, DecideToWithdraw(p_1, s(p_1))\rangle \xrightarrow{Withdraw(p_1)} \langle\langle \text{End}, p_1\rangle, B_e - p_1, B_c\rangle$

$\langle\langle \text{DecisionAnswer}, p_1, s(p_1)\rangle, DecideToOffic(p_1, s(p_1))\rangle \xrightarrow{Offic(p_1)} \langle\langle \text{AgreementAnswer}, p_1\rangle, NOP\rangle$

Reglas del intérprete correspondiente al prococolo de Sian para más de dos agentes:

$\langle\langle \text{Init}, p_1 \in B_e\rangle, WantToPropose(p_1)\rangle \xrightarrow{bc(Propose(p_1,myid))} \langle\langle \text{OpinionPropose}, p_1, myid, N-1, []\rangle, NOP\rangle$

$\langle\langle \text{Init}, p_1 \in B_e\rangle, true\rangle \xrightarrow[Propose(p_1,id)]{} \langle\langle \text{OpinionAnswer}, p_1, id\rangle, NOP\rangle$

$\langle\langle \text{Opinion}, p_1, myid, Succ(n), l\rangle, true\rangle \xrightarrow[Confirm(p_1)]{} \langle\langle \text{OpinionPropose}, p, myid, n, Confirm(p_1) :: l\rangle, NOP\rangle$

$\langle\langle \text{Opinion}, p_1, myid, Succ(n), l\rangle, true\rangle \xrightarrow[Disagree(p_1)]{} \langle\langle \text{OpinionPropose}, p, myid, n, Disagree(p_1) :: l\rangle, NOP\rangle$

$\langle\langle \text{Opinion}, p_1, myid, Succ(n), l\rangle, true\rangle \xrightarrow[NoOpinion(p_1)]{} \langle\langle \text{OpinionPropose}, p, myid, n, NoOpinion(p_1) :: l\rangle, NOP\rangle$

$\langle\langle \text{Opinion}, p_1, myid, Succ(n), l\rangle, true\rangle \xrightarrow[Modify(p_1,p_2,id)]{} \langle\langle \text{OpinionPropose}, p_1, myid, n, Modify(p_1, p_2) :: l\rangle, NOP\rangle$

$\langle\langle \text{Opinion}, p_1, myid, Zero, l\rangle, Modify(p, q)in\rangle \xrightarrow{Confirm(p_1)} \langle\langle \text{OpinionPropose}, p, myid, n, Confirm(p_1) :: l\rangle, NOP\rangle$

$\langle\langle \text{Opinion}, p_1, myid, Succ(n), l\rangle, true\rangle \xrightarrow[Disagree(p_1)]{} \langle\langle \text{OpinionPropose}, p, myid, n, Disagree(p_1) :: l\rangle, NOP\rangle$

$\langle\langle \text{Opinion}, p_1, myid, Succ(n), l\rangle, true\rangle \xrightarrow[NoOpinion(p_1)]{} \langle\langle \text{OpinionPropose}, p, myid, n, NoOpinion(p_1) :: l\rangle, NOP\rangle$

$\langle\langle \text{Opinion}, p_1, myid, Succ(n), l\rangle, Modify(p_1, p_2, id) \in l\rangle \xrightarrow{Signal(p_1,p_2,id)} \langle\langle \text{OpinionAnswer}, p_2, id\rangle, NOP\rangle$

$\langle\langle \text{Opinion}, p_1, myid, Succ(n), l\rangle, Modify(p_1, p_2, id) \notin l \wedge WTOffic\rangle \xrightarrow{bc(Offic(p_1))} \langle\langle \text{OpinionAnswer}, p_2, id\rangle, NOP\rangle$

$\langle\langle \text{Opinion}, p_1\rangle, timeout\rangle \overrightarrow{\varepsilon} \langle\langle \text{End}, p_1\rangle, \text{NOP}\rangle$

$\langle\langle \text{Init}, p_1 \in B_e\rangle, WantToAssert(p_1)\rangle \xrightarrow{Assert(p_1)} \langle\langle \text{AgreementPropose}, p_1\rangle, NOP\rangle$

$\langle\langle \text{AgreementPropose}, p_1\rangle, true\rangle \xrightarrow[Accept(p_1)]{} \langle\langle \text{End}, p_1\rangle, B_e - p_1, B_c + p_1\rangle$

$\langle\langle \text{AgreementPropose}, p_1\rangle, timeout\rangle \overrightarrow{\varepsilon} \langle\langle \text{End}, p_1\rangle, \text{NOP}\rangle$

$\langle\langle \text{Opinion}, p_1\rangle, true\rangle \xrightarrow[p_1, Confirm(p_1)]{} \langle\langle \text{DecisionPropose}, Confirm(p_1)\rangle, NOP\rangle$

$\langle\langle \text{Opinion}, p_1\rangle, true\rangle \xrightarrow[p_1, Disagree(p_1)]{} \langle\langle \text{DecisionPropose}, Disagree(p_1)\rangle, NOP\rangle$

$\langle\langle \text{Opinion}, p_1\rangle, true\rangle \xrightarrow[p_1, NoOpinion(p_1)]{} \langle\langle \text{DecisionPropose}, NoOpinion(p_1)\rangle, NOP\rangle$

$\langle\langle \text{Opinion}, p_1\rangle, true\rangle \xrightarrow[Modify(p_1,p_2)]{} \langle\langle \text{CounterOpinion}, p_2\rangle, B_e - p_1, B_c\rangle$

$\langle\langle \text{Opinion}, p_1\rangle, timeout\rangle \overrightarrow{\varepsilon} \langle\langle \text{End}, p_1\rangle, \text{NOP}\rangle$

$\langle\langle \text{DecisionPropose}, p_1, s(p_1)\rangle, DecideToWithdraw(p_1, s(p_1))\rangle \xrightarrow{Withdraw(p_1)} \langle\langle \text{End}, p_1\rangle, B_e - p_1, B_c\rangle$

$\langle\langle \text{DecisionPropose}, p_1, s(p_1)\rangle, DecideToOffic(p_1, s(p_1))\rangle \xrightarrow{Offic(p_1)} \langle\langle \text{AgreementPropose}, p_1\rangle, NOP\rangle$

$\langle\langle \text{DecisionAnswer}, p_1\rangle, true\rangle \xrightarrow[Withdraw(p_1)]{} \langle\langle \text{End}, p_1\rangle, B_e - p_1, B_c\rangle$

$\langle\langle \text{DecisionAnswer}, p_1\rangle, true\rangle \xrightarrow[Offic(p_1)]{} \langle\langle \text{AgreementAnswer}, p_1\rangle, NOP\rangle$

$\langle\langle \text{DecisionAnswer}, p_1\rangle, timeout\rangle \overrightarrow{\varepsilon} \langle\langle \text{End}, p_1\rangle, \text{NOP}\rangle$

$\langle\langle \text{AgreementAnswer}, p_1\rangle, true\rangle \xrightarrow{Accept(p_1)} \langle\langle \text{End}, p_1\rangle, B_e - p_1, B_c + p_1\rangle$

$\langle\langle \text{CounterOpinion}, p_1\rangle, WantToConfirm(p_1)\rangle \xrightarrow{Confirm(p_1)} \langle\langle \text{DecisionAnswer}, p_1\rangle, NOP\rangle$

$\langle\langle \text{CounterOpinion}, p_1\rangle, WantToDisagree(p_1)\rangle \xrightarrow{Disagree(p_1)} \langle\langle \text{DecisionAnswer}, p_1\rangle, NOP\rangle$

$\langle\langle \text{CounterOpinion}, p_1\rangle, NoOpinion(p_1)\rangle \xrightarrow{NoOpinion(p_1)} \langle\langle \text{DecisionAnswer}, p_1\rangle, NOP\rangle$

$\langle\langle \text{CounterOpinion}, p_1\rangle, WantToModify(p_1, p_2)\rangle \xrightarrow{Modify(p_1,p_2)} \langle\langle \text{Opinion}, p_2\rangle, B_e - p, B_c\rangle$

Apéndice D

Conceptualización DL de una holarquía

Índice General

D.1. Especificación en lógica descriptiva de un caso de aplicación 281
D.2. Especificación OWL del modelo de holarquía 295

En la primera sección de este apéndice se muestra la formalización en lógica descriptiva del modelo de organización holónico y orientado a roles recogido en la figura D.1 para, a continuación, mostrar este mismo modelo descrito en la sintaxis OWL-DL.

D.1. Especificación en lógica descriptiva del caso de aplicación: Gestión holónica de los aspectos de seguridad de un SLA

Las siguientes secciones recogen la formalización en lógica descriptiva del modelo de organización holónico y orientado a roles descrito en la figura D.1. Para ello se ha utilizado una sintaxis concreta basada en LISP, en lugar de la sintaxis abstracta utilizada a lo largo del resto de la tesis. El propósito es mostrar el aspecto típico de la entrada a una herramienta de razonamiento deductivo basado en lógica descriptiva como puede ser RACER [HM01]. El cuadro D.1 recoge la relación entre ambos tipos de sintaxis.

La primera subsección muestra una base de conocimiento terminológico (TBox) con la semántica de dicho modelo, así como una base de conocimiento asertivo (ABox) correspondiente a la instanciación de una región del caso práctico propuesto

	Sintaxis Concreta	**Sintaxis DL**
Negación	(not C)	$\neg C$
Disyunción	(or $C_1 \cdots C_n$)	$C_1 \sqcup \cdots \sqcup C_n$
Conjunción	(and $C_1 \cdots C_n$)	$C_1 \sqcap \cdots \sqcap C_n$
Restricción de valor	(all P C)	$\forall P.C$
Restricción en existencia	(some P C)	$\exists P.C$
Restricción en cardinalidad no calificada	(at-least n P) (at-most n P) (exactly n P)	$\langle \geq nP \rangle$ $\langle \leq nP \rangle$ $\langle = nP \rangle$
Restricción en cardinalidad calificada	(at-least n P C) (at-most n P C) (exactly n P C)	$\langle \geq nP.C \rangle$ $\langle \leq nP.C \rangle$ $\langle = nP.C \rangle$
Propiedad funcional	(P :feature t)	$\langle \leq 1P \rangle$
Inclusión de conceptos	(implies C_1 C_2)	$C_1 \sqsubseteq C_2$
Subpropiedad	(Q :parents $P_1 \cdots P_n$)	$Q \sqsubseteq P_1$ ... $Q \sqsubseteq P_n$
Propiedad simétrica	(P :symmetric t)	$P \equiv P^-$
Propiedad transitiva	(P :transitive t)	$P \equiv P \circ^i P$,i=1..n
Equivalencia de conceptos	(equivalent C_1 C_2	$C_1 \equiv C_2$
Conceptos disjuntos	(disjoint $C_1 \cdots c_n$)	$C_i \sqsubseteq \neg C_{i+1} \sqcap \cdots \sqcap \neg C_n$, i=1..n
Propiedad inversa	(P :inverse Q)	$P \equiv Q^-$

Cuadro D.1: Correspondencia entre la sintaxis LISP y la sintaxis abstracta DL

en la sección 5.5. La segunda subsección muestra una base de conocimiento terminológico (TBox) en la que el modelo descrito en la figura D.1 se contempla realmente como un meta-modelo, permitiendo con ello su instanciación en diferentes modelos. Esta solución resulta más flexible a la hora de modelar organizaciones de gran tamaño organizadas en torno a unidades estructurales que presentan configuraciones similares, si no idénticas, en términos de roles y relaciones entre éstos. Así, cada modelo, instancia del metamodelo recogido en la figura 5.2, puede instanciarse a su vez sucesivas veces para representar cada una de estas unidades funcionales. La sección 5.5 presenta un caso de aplicación que refleja esta necesidad. Este caso muestra además como esta última aproximación permite mayor expresividad a la hora de, por ejemplo, establecer el carácter exclusivo (unicidad) de un rol (un rol único en una unidad organizativa no tiene por qué ser único en todas las instanciaciones de dicha unidad). El hecho de que una base de conocimiento terminológico no pueda expresar el nivel de meta-modelado, ha obligado a especificar el modelo de organización concreto como una restricción (en lugar de una instancia) del (meta)modelo de organización holónico en términos de constructores y axiomas de la lógica descriptiva.

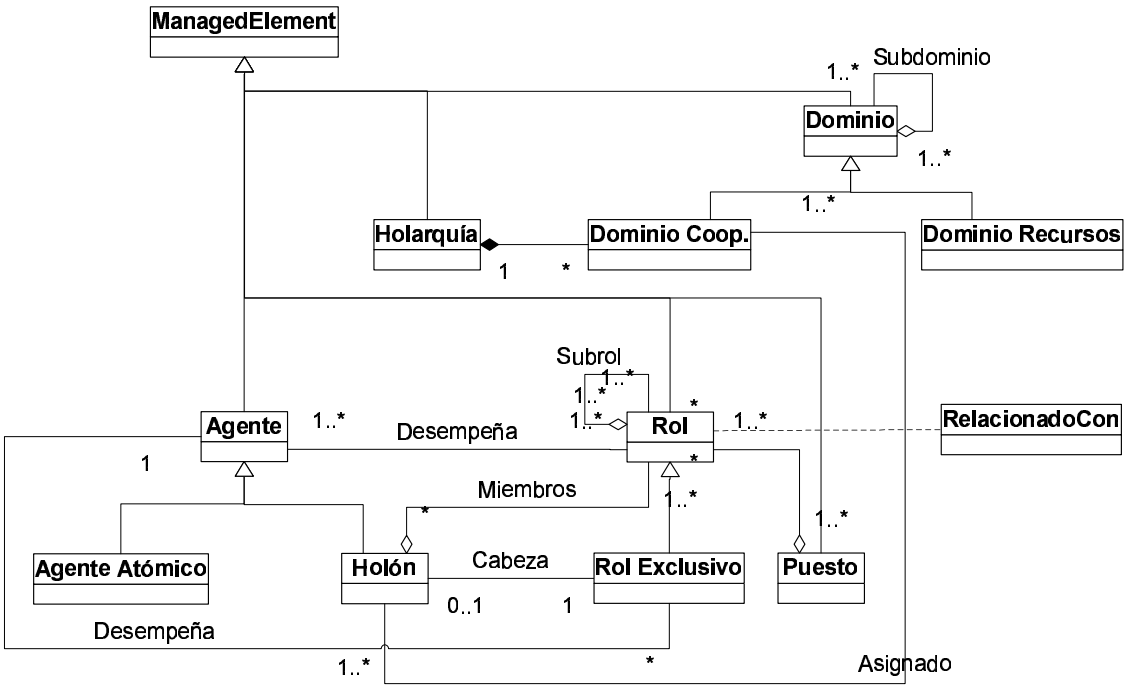

Figura D.1: Composición de una holarquía orientada a roles

D.1.1. Modelo de organización holónico

Se describe a continuación el modelo de organización para el caso de aplicación *Gestión holónica de los aspectos de los aspectos de seguridad de un SLA* presentado en la sección 5.5.

```
;;;=========================================================
;;;Modelo de organización para el caso de aplicación:
;;;   Gestión holónica de los aspectos de seguridad de un SLA
;;;=========================================================
;;; Archivo Organizacion.racer

(in-knowledge-base MOrganizativo)

;;; Signatura para el TBox

(signature
   :atomic-concepts (
                     ManagedElement Agent AtomicAgent Holon UniqueRole Role
                     CooperativeDomain ResourcesDomain Domain Holarchy Job
                     RelatedTo EventExpression InteractionProtocol Norm
                     Obligation Violation Interdiction Permission
                     Authorisation PositiveAuth NegativeAuth Task
                    )
   :roles (
           (SubDomainOf :transitive t :domain Domain :range Domain)
           (HasHead :inverse HeadOf :domain Holon
                    :range UniqueRole :feature t)
           (Plays :inverse PlayedBy :domain Agent :range Role)
           (ConsistOf :domain Job :range Role)
           (Contains :domain Holarchy :range CooperativeDomain)
           (AssignedTo :domain Holon :range CooperativeDomain)
           (HasMember :domain Holon :range Role)
           (RelatedTo_Antecedent :domain RelatedTo :range Role :feature t)
           (RelatedTo_Dependent :domain RelatedTo :range Role :feature t)
           (fromSubject :feature t :domain (or Norm
                                               Authorisation
                                               Violation)
                                :range (or Role CooperativeDomain)
           (toTarget :feature t :domain (or Norm
                                            Authorisation
                                            Violation)
                                :range (or Role CooperativeDomain))
           (forTask :domain (or Norm Authorisation) :range Task)
           (onTrigger :feature t :domain Norm
                                 :range EventExpression)
           (onCondition :feature t :domain Authorisation
                                   :range ConditionExpression)
           (onViolation :domain Obligation :range Violation)
           (compensates :domain Violation :range Norm)
           (acceptProtocol :domain Role :range InteractionProtocol)
           (hasPolicy :domain Role :range (or Norm Authorisation))
           (impliesProtocol :domain RelatedTo :range InteractionProtocol)
           (impliesPolicy :domain RelatedTo :range (or Norm Authorisation))
          )
```

```
   :instances (InspectorSeguridad GestorArchivo GestorEvaluacion
               GestorLog Recolector NotificarAnomalia NotificarAtaque
               EstructurarLog Inspeccionar Recopilar DeptContabilidad
               DeptProduccion DeptAdministracion DeptIyD
               DeptInformatica DeptComercial HolonSQA LocalNetworkSite
               Region PoliticaNotifAtaque PInotificacion
               PoliticaNotifAnomalia PInotificacion PoliticaEstructLog
               PInotificacion PoliticaInspeccion PIinspeccion
               PoliticaRecopilacion PIrecopilacion AgenteInsp AgenteInv
               AgenteExpertoLog AgenteRecolector
               AgenteInspGeneral AgenteGestorEval AgenteAdminArchivo
              )
)

;;TBox

;;Modelo de organización

(implies Role (and ManagedElement
                   (all PlayedBy Agent)
                   (at-least 1 PlayedBy)
                   (all (inv RelatedTo_Antecedent) RelatedTo)
                   (all (inv RelatedTo_Dependent) RelatedTo)
              )
)
(implies RelatedTo (and (some RelatedTo_Antecedent Role)
                        (at-most 1 RelatedTo_Antecedent)
                        (some RelatedTo_Dependent Role)
                        (at-most 1 RelatedTo_Dependent)
                   )
)
(implies UniqueRole (and Role
                         (some PlayedBy Agent)
                         (exactly 1 PlayedBy)
                         (all HeadOf Holon)
                    )
)
(implies Job (and ManagedElement
                  (all ConsistOf Role)
                  (at-least 2 ConsistOf)
             )
)
(implies Domain (and ManagedElement
                     (all SubDomainOf Domain)
                )
)
(implies CooperativeDomain Domain)
(implies ResourcesDomain Domain)
(disjoint CooperativeDomain ResourcesDomain)
(implies Holarchy (and ManagedElement
                       (all Contains CooperativeDomain)
                       (at-least 1 Contains)
                  )
)
(implies Agent (and ManagedElement
                    (all Plays Role)
```

```
                    (at-least 1 Plays)
               )
)
(implies AtomicAgent Agent)
(disjoint AtomicAgent Holon)
(implies Agent (or AtomicAgent Holon))
(implies Holon (and Agent
                    (some AssignedTo CooperativeDomain)
                    (exactly 1 AssignedTo)
                    (all HasMember Role)
                    (at-least 1 HasMember)
                    (some HasHead UniqueRole)
                    (at-most 1 HasHead)
               )
)

;;;Submodelo normativo

(implies Norm (and (some fromSubject (or Role
                                        CooperativeDomain))
                   (at-most 1 fromSubject)
                   (some toTarget (or Role
                                      Domain))
                   (at-most 1 toTarget)
                   (all forTask Task)
                   (at-least 1 forTask Task)
                   (some onTrigger EventExpression)
                   (at-most 1 onTrigger)
               )
)
(implies Obligation (and Norm
                         (all onViolation Violation)
                         (at-least 1 onViolation)
                     )
)
(implies Violation (and (some fromSubject (or Role
                                             CooperativeDomain))
                        (at-most 1 fromSubject)
                        (some toTarget (or Role
                                           Domain))
                        (at-most 1 toTarget)
                        (all compensates Task)
                        (at-least 1 compensates)
                        (some (inv onViolation) Obligation)
                        (at-most 1 (inv onViolation))
                   )
)
(implies Permission Obligation)
(implies Interdiction (and Obligation (not Permission)))
(implies Authorisation (and (some fromSubject (or Role
                                        CooperativeDomain))
                            (at-most 1 fromSubject)
                            (some toTarget (or Role
                                               Domain))
                            (at-most 1 toTarget)
                            (all forTask Task)
```

```
                          (at-least 1 forTask Task)
                          (some onCondition ConditionExpression)
                          (at-most 1 onCondition)
                     )
)
(implies PositiveAuth Authorisation)
(implies NegativeAuth (and Authorisation (not PositiveAuth)))
(implies Role (and (all acceptProtocol InteractionProtocol)
                   (at-least 1 acceptProtocol)
                   (all hasPolicy (or Norm
                                      Authorisation))
              )
)
(implies RelatedTo (and (all impliesProtocol InteractionProtocol)
                        (at-least 1 impliesProtocol)
                        (all impliesPolicy (or Norm
                                               Authorisation)
                                           )
                   )
)

;; ABox

(instance RegionDeptContabilidad CooperativeDomain)
(instance RegionDeptProduccion CooperativeDomain)
(instance RegionDeptAdministracion CooperativeDomain)
(instance RegionDeptIyD CooperativeDomain)
(instance RegionDeptInformatica CooperativeDomain)
(instance RegionDeptComercial CooperativeDomain)
(related DeptInformatica DeptIyD SubDomainOf)
(instance InspectorSeguridad uniqueRole)
(instance InspectorSeguridadGeneral UniqueRole)
(related InspectorSeguridad InspectorSeguridadGeneral UniqueRole)
(instance GestorArchivo Role)
(instance GestorEvaluacion Role)
(instance GestorLog Role)
(instance Recolector Role)
(instance NotificarAnomalia RelatedTo)
(instance NotificarAtaque RelatedTo)
(instance EstructurarLog RelatedTo)
(instance Inspeccionar RelatedTo)
(related NotificarAnomalia GestorArchivo RelatedTo_antecedent)
(related NotificarAnomalia InspectorSeguridad RelatedTo_dependent)
(related NotificarAtaque InspectorSeguridad RelatedTo_antecedent)
(related NotificarAtaque GestorEvaluacion RelatedTo_dependent)
(related EstructurarLog InspectorSeguridad RelatedTo_antecedent)
(related EstructurarLog GestorLog RelatedTo_dependent)
(related Inspeccionar InspectorSeguridad RelatedTo_antecedent)
(related Inseccionar Recolector RelatedTo_dependent)
(instance HolonSQA Holon)
(related HolonSQA LocalNetworkSite AssignedTo)
(instance HolonGlobal Holon)
(related HolonGlobal Region AssignedTo)
(related LocalNetworkSite Region SubdomainOf)
(related Holarquia Region Contains)
(related HolonSQA InspectorSeguridadGeneral HasMember)
```

```
(related HolonSQA GestorArchivo HasMember)
(related HolonSQA GestorEvaluacion HasMember)
(related HolonSQA InspectorSeguridad HasMember)
(related HolonSQA GestorLog HasMember)
(related HolonSQA Recolector HasMember)
(related HolonSQA LocalNetworkSite AssignedTo)
(related NotificarAtaque PoliticaNotifAtaque impliesPolicy)
(related NotificarAtaque PInotificacion impliesProtocol)
(related NotificarAnomalia PoliticaNotifAnomalia impliesPolicy)
(related NotificarAnomalia PInotificacion impliesProtocol)
(related EstructurarLog PoliticaEstructLog impliesPolicy)
(related EstructurarLog PInotificacion impliesProtocol)
(related Inspeccionar PoliticaInspeccion impliesPolicy)
(related Inspeccionar PIinspeccion impliesProtocol)
(instance AgenteInsp AtomicAgent)
(instance AgenteExpertoLog  AtomicAgent)
(instance AgenteExpertoLog2 AtomicAgent)
(instance AgenteRecolector AtomicAgent)
(instance AgenteInspGeneral AtomicAgent)
(instance AgenteGestorEval AtomicAgent)
(instance AgenteAdminArchivo AtomicAgent)
(related AgenteInsp InspectorSeguridad plays)
(related AgenteInv GestorInventario plays)
(related AgenteExpertoLog GestorLog plays)
(related AgenteExpertoLog2 GestorLog plays)
(related AgenteRecolector Recolector plays)
(related AgenteInspGeneral InspectorSeguridadGeneral plays)
(related AgenteGestorEval GestorEvaluacion plays)
(related AgenteAdminArchivo GestorArchivo plays)
```

D.1.2. Metamodelo de organización holónico

```
;;;====================
;;;Modelo Organizativo
;;;====================
;;; Archivo Organizacion.racer

(in-knowledge-base MOrganizativo)

;;; Signatura para el TBox

(signature
   :atomic-concepts (
           ManagedElement Agent AtomicAgent Holon
           UniqueRole Role CooperativeDomain ResourcesDomain
           Domain Holarchy Job RelatedTo EventExpression
           InteractionProtocol Norm Obligation Violation
           Interdiction Permission Authorisation
           PositiveAuth NegativeAuth Task HolarquiaSQA
           LocalNetworkSite GlobalRegionNetwork DeptInformatica
           DeptIyD InspectorSeguridadGlobal InspectorSeguridad
           GestorArchivo GestorEvaluacion GestorLog
           Recolector NotificarAnomalia NotificarAtaque
           EstructurarLog Inspeccionar Recopilar HolonSQA
           HolonSQAGlobal HolonColector AgenteInsp AgenteInv
```

```
          AgenteExpertoLog AgenteRecolector AgenteInspGeneral
          AgenteGestorEval AgenteAdminArchivo Politica
          NotifAnomalia PInotificacion PoliticaEstructLog
          PoliticaInspeccion PIinspeccion PoliticaRecopilacion
          PIrecopilacion PoliticaNotifAtaque PoliticaNotifAnomalia
          HolonAdministracion
                  )
  :roles (
          (SubDomainOf  :transitive t :domain Domain :range Domain)
          (HasHead      :inverse HeadOf :domain Holon
                        :range UniqueRole :feature t)
          (Plays        :inverse PlayedBy :domain Agent :range Role)
          (ConsistOf    :domain Job :range Role)
          (Contains     :inverse ContainedIn
                        :domain Holarchy :range  CooperativeDomain)
          (AssignedTo   :inverse HasAssigned
                        :domain Holon :range CooperativeDomain)
          (HasMember    :inverse MemberOf
                        :domain Holon :range Role)
          (RelatedTo_Antecedent :domain RelatedTo
                                :range Role :feature t)
          (RelatedTo_Dependent  :domain RelatedTo
                                :range Role :feature t)
          (SubRoleOf    :transitive t :domain Role :range Role)
          (fromSubject  :feature t :domain Norm
                        :range (or Role CooperativeDomain))
          (toTarget     :feature t :domain Norm
                        :range (or Role Domain))
          (forTask      :domain Norm :range Task)
          (onTrigger    :feature t :domain Norm
                        :range EventExpression)
          (onViolation  :domain Obligation :range Violation)
          (compensates  :domain Violation
                        :range (or Norm NegativeAuth))
          (acceptProtocol  :domain Role
                           :range InteractionProtocol)
          (hasPolicy       :domain Role
                           :range (or Norm Authorisation))
          (impliesProtocol :domain RelatedTo
                           :range InteractionProtocol)
          (impliesPolicy   :domain RelatedTo
                           :range (or Norm Authorisation))
          (NotificarAnomalia_Antecedent :parent RelatedTo_Antecedent)
          (NotificarAnomalia_Dependent  :parent RelatedTo_Dependent)
          (NotificarAtaque_Antecedent   :parent RelatedTo_Antecedent)
          (NotificarAtaque_Dependent    :parent RelatedTo_Dependent)
          (EstructurarLog_Antecedent    :parent RelatedTo_Antecedent)
          (EstructurarLog_Dependent     :parent RelatedTo_Dependent)
          (Inspeccionar_Antecedent      :parent RelatedTo_Antecedent)
          (Inspeccionar_Dependent       :parent RelatedTo_Dependent)
         )
)

;;TBox

(implies Role (and ManagedElement
```

```
                    (all PlayedBy Agent)
                    (at-least 1 PlayedBy)
                    (all (inv RelatedTo_Antecedent) RelatedTo)
                    (all (inv RelatedTo_Dependent) RelatedTo)
               )
)
(implies RelatedTo (and (some RelatedTo_Antecedent Role)
                        (at-most 1 RelatedTo_Antecedent)
                        (some RelatedTo_Dependent Role)
                        (at-most 1 RelatedTo_Dependent)
                    )
)
(implies UniqueRole (and Role
                         (some PlayedBy Agent)
                         (exactly 1 PlayedBy)
                         (all HeadOf Holon)
                     )
)
(implies Job (and ManagedElement
                  (all ConsistOf Role)
                  (at-least 2 ConsistOf)
             )
)
(implies Domain (and ManagedElement
                     (all SubDomainOf Domain)
                )
)
(implies CooperativeDomain Domain)
(implies ResourcesDomain Domain)
(disjoint CooperativeDomain ResourcesDomain)
(implies Holarchy (and ManagedElement
                       (all Contains CooperativeDomain)
                       (at-least 1 Contains)
                  )
)

(implies Agent (and ManagedElement
                    (all Plays Role)
                    (at-least 1 Plays)
               )
)
(implies AtomicAgent Agent)
(disjoint AtomicAgent Holon)
(implies Agent (or AtomicAgent Holon))
(implies Holon (and Agent
                    (some AssignedTo CooperativeDomain)
                    (exactly 1 AssignedTo)
                    (all HasMember Role)
                    (at-least 1 HasMember)
                    (some HasHead UniqueRole)
                    (at-most 1 HasHead)
               )
)

;;Submodelo normativo
```

```
(implies Norm (and (some fromSubject (or Role
                                         CooperativeDomain
                                     )
                   )
                   (at-most 1 fromSubject)
                   (some toTarget (or Role
                                      Domain
                                  )
                   )
                   (at-most 1 toTarget)
                   (all forTask Task)
                   (at-least 1 forTask Task)
                   (some onTrigger EventExpression)
                   (at-most 1 onTrigger)
               )
)
(implies Obligation (and Norm
                         (all onViolation Violation)
                         (at-least 1 onViolation)
                    )
)
(implies Violation (and (some fromSubject (or Role
                                              CooperativeDomain
                                          )
                        )
                        (at-most 1 fromSubject)
                        (some toTarget (or Role
                                           Domain
                                       )
                        )
                        (at-most 1 toTarget)
                        (all compensates (or Norm
                                             NegativeAuth
                                         )
                        )
                        (at-least 1 compensates)
                        (some (inv onViolation) Norm)
                        (at-most 1 (inv onViolation))
                   )
)
(implies Permission Obligation)
(implies Interdiction (and Obligation (not Permission)))
(implies PositiveAuth Authorisation)
(implies NegativeAuth (and Authorisation (not PositiveAuth)))
(implies Role (and (all acceptProtocol InteractionProtocol)
                   (at-least 1 acceptProtocol)
                   (all hasPolicy (or Norm
                                      Authorisation
                                  )
                   )
               )
)
(implies RelatedTo (and (all impliesProtocol InteractionProtocol)
                        (at-least 1 impliesProtocol)
                        (all impliesPolicy (or Norm
                                               Authorisation
```

```
                                          )
                        )
                    )
)
(implies HolarquiaSQA (and Holarchy
                           (all contains (or LocalNetworkSite
                                             GlobalRegionNetwork
                                         )
                           )
                       )
)
(implies LocalNetworkSite (and CooperativeDomain
                               (all HasAssigned HolonSQA)
                               (some ContainedIn HolarquiaSQA)
                               (at-most 1 ContainedIn)
                               (some SubDomainOf GlobalRegionNetwork)
                               (at-most 1 SubDomainOf)
                           )
)
(implies GlobalRegionNetwork (and CooperativeDomain
                                  (all HasAssigned (or HolonSQAGlobal
                                                       GestorEvaluacion
                                                   )
                                  )
                                  (some ContainedIn HolarquiaSQA)
                                  (at-most 1 ContainedIn)
                              )
)
(implies InspectorSeguridadGlobal
           (and InspectorSeguridad
                (some (inv NotificarAnomalia_Dependent) NotificarAnomalia)
                (at-most 1 (inv NotificarAnomalia_Dependent))
                (some (inv NotificarAtaque_Antecedent) NotificarAtaque)
                (at-most 1 (inv NotificarAtaque_Antecedent))
           )
)
(implies InspectorSeguridad
           (and UniqueRole
                (some (inv NotificarAtaque_Antecedent) NotificarAtaque)
                (at-most 1 (inv NotificarAtaque_Antecedent))
                (some (inv EstructurarLog_Antecedent) EstructurarLog)
                (at-most 1 (inv EstructurarLog_Antecedent))
                (some (inv Inspeccionar_Antecedent) Inspeccionar)
                (at-most 1 (inv Inspeccionar_Antecedent))
           )
)
(implies GestorArchivo
           (and Role
                (some (inv NotificarAnomalia_Antecedent) NotificarAnomalia)
                (at-most 1 (inv NotificarAnomalia_Antecedent))
           )
)
(implies GestorEvaluacion
           (and UniqueRole
                (some (inv NotificarAtaque_Dependent) NotificarAtaque)
                (at-most 1 (inv NotificarAtaque_Dependent))
```

```
          )
)
(implies GestorLog
          (and Role
               (some (inv EstructurarLog_Dependent) EstructurarLog)
                              (at-most 1 (inv EstructurarLog_Dependent))
          )
)
(implies Recolector
          (and UniqueRole
               (some (inv Inspeccionar_Dependent) Inspeccionar)
               (at-most 1 (inv Inspeccionar_Dependent))
          )
)
(implies NotificarAnomalia
          (and (some NotificarAnomalia_Antecedent GestorArchivo)
               (at-most 1 NotificarAnomalia_Antecedent)
               (some NotificarAnomalia_Dependent InspectorSeguridadGlobal)
               (at-most 1 NotificarAnomalia_Dependent)
               (some impliesPolicy PoliticaNotifAnomalia)
               (at-most 1 impliesPolicy)
               (some impliesProtocol PInotificacion)
               (at-most 1 impliesProtocol)
           )
)
(implies NotificarAtaque
           (and RelatedTo
                (some NotificarAtaque_Antecedent (or InspectorSeguridadGlobal
                                                     InspectorSeguridad
                                                 )
                )
                (at-most 1 NotificarAtaque_Antecedent)
                (some NotificarAtaque_Dependent GestorEvaluacion)
                (at-most 1 NotificarAtaque_Dependent)
                (some impliesPolicy PoliticaNotifAtaque)
                (at-most 1 impliesPolicy)
                (some impliesProtocol PInotificacion)
                (at-most 1 impliesProtocol)
           )
)
(implies EstructurarLog
           (and RelatedTo (some EstructurarLog_Antecedent InspectorSeguridad)
                (at-most 1 EstructurarLog_Antecedent)
                (some EstructurarLog_Dependent GestorLog)
                (at-most 1 EstructurarLog_Dependent)
                (some impliesPolicy PoliticaEstructLog)
                (at-most 1 impliesPolicy)
                (some impliesProtocol PInotificacion)
                (at-most 1 impliesProtocol)
           )
)
(implies Inspeccionar
           (and RelatedTo
                (some Inspeccionar_Antecedent InspectorSeguridad)
                (at-most 1 Inspeccionar_Antecedent)
                (some Inspeccionar_Dependent Recolector)
```

```
                (at-most 1 Inspeccionar_Dependent)
                (some impliesPolicy PoliticaInspeccion)
                (at-most 1 impliesPolicy)
                (some impliesProtocol PIinspeccion)
                (at-most 1 impliesProtocol)
            )
)
(implies HolonSQA (and Holon
                      (all HasMember (or InspectorSeguridad
                                         Recolector
                                         GestorLog
                                     )
                      )
                      (some AssignedTo LocalNetworkSite)
                      (at-most 1 AssignedTo)
                      (some HasHead InspectorSeguridad)
                  )
)
(implies HolonSQAGlobal (and Holon
                             (all HasMember (or
                                              InspectorSeguridadGlobal
                                              GestorEvaluacion
                                              GestorArchivo
                                            )
                             )
                             (some AssignedTo GlobalRegionNetwork)
                             (at-most 1 AssignedTo)
                             (some HasHead GestorEvaluacion)
                         )
)
(implies AgenteInsp (and AtomicAgent
                         (some Plays InspectorSeguridad)
                    )
)
(implies AgenteExpertoLog (and AtomicAgent
                               (some Plays GestorLog)
                          )
)
(implies AgenteRecolector (and AtomicAgent
                               (some Plays Recolector)
                          )
)
(implies AgenteInspGeneral (and AtomicAgent
                                (some Plays InspectorSeguridadGlobal)
                           )
)
(implies AgenteGestorEval (and AtomicAgent
                               (some Plays GestorEvaluacion)
                          )
)
(implies AgenteAdminArchivo (and AtomicAgent
                                 (some Plays GestorArchivo)
                            )
)
(implies PIInspeccion InteractionProtocol)
(implies PINotificacion InteractionProtocol)
```

```
(implies PIRecopilacion InteractionProtocol)
(implies PoliticaEstructLog Obligation)
(implies PoliticaInspeccion Obligation)
(implies PoliticaNotifAnomalia Obligation)
(implies PoliticaNotifAtaque Obligation)
(implies PoliticaRecopilacion Obligation)
```

D.2. Especificación OWL del modelo de holarquía

```
<?xml version="1.0" encoding="UTF-8"?> <!DOCTYPE owl [
   <!ENTITY hm "http://jupiter.ls.fi.upm.es/OWL_HolonicModel.rdf#">
   <!ENTITY owl "http://www.w3.org/2002/07/owl#">
   <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">
   <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
   <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#">
]>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:owl="http://www.w3.org/2002/07/owl#"
xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
xmlns="http://jupiter.ls.fi.upm.es/HolonicModel.rdf#"
xmlns:dc="http://purl.org/dc/elements/1.1/">
   <owl:Ontology rdf:about="http://jupiter.ls.fi.upm.es/HolonicModel.rdf#">
      <rdfs:comment>Ontologia para un modelo holarquico</rdfs:comment>
      <owl:versionInfo>
            $Id: HolonicModelOntology v 1.0 02/02/2003 Javier Soriano $
        </owl:versionInfo>
      <owl:imports rdf:resource="http://www.w3.org/2002/07/owl"/>
      <owl:imports rdf:resource="http://www.w3.org/2000/01/rdf-schema"/>
      <dc:title>OWL Holonic Model Ontology</dc:title>
      <dc:creator>Francisco Javier Soriano Camino</dc:creator>
      <dc:subject>OWL; Ontologies; Holon; Holarchy</dc:subject>
      <dc:description>Ontologia OWL para el modelo de holarquica</dc:description>
      <dc:publisher>Grupo de Redes y Sistemas Distribuidos. Facultad de Informatica</dc:publisher>
      <dc:date>Feb 02, 2003</dc:date>
      <dc:format>text/xml</dc:format>
      <dc:language>es</dc:language>
      <dc:identifier>urn:TIC2001-3451:ontologies:holonicModel</dc:identifier>
   </owl:Ontology>

  <owl:Class rdf:ID="ManagedElement">
   </owl:Class>

   <owl:Class rdf:ID="Role">
      <rdfs:subClassOf rdf:resource="#ManagedElement"/>
   </owl:Class>

   <owl:Class rdf:ID="UniqueRole">
      <rdfs:subClassOf rdf:resource="#Role"/>
   </owl:Class>

   <owl:Class rdf:ID="Job">
      <rdfs:subClassOf rdf:resource="#ManagedElement"/>
   </owl:Class>

   <owl:Class rdf:ID="CooperativeDomain">
      <rdfs:subClassOf rdf:resource="#Domain"/>
      <owl:disjointWith rdf:resource="#ResourcesDomain"/>
   </owl:Class>

   <owl:Class rdf:ID="ResourcesDomain">
      <rdfs:subClassOf rdf:resource="#Domain"/>
   </owl:Class>

   <owl:Class rdf:ID="Holarchy">
      <rdfs:subClassOf rdf:resource="#ManagedElement"/>
```

```
</owl:Class>

<owl:Class rdf:ID="Agent">
   <rdfs:subClassOf rdf:resource="#ManagedElement"/>
 </owl:Class>

<owl:Class rdf:ID="AtomicAgent">
   <rdfs:subClassOf rdf:resource="#Agent"/>
   <owl:disjointWith rdf:resource="#Holon"/>
</owl:Class>

<owl:Class rdf:ID="Holon">
   <rdfs:subClassOf rdf:resource="#Agent"/>
</owl:Class>

<owl:Class rdf:about="#Role">
   <rdfs:subClassOf>
      <owl:Restriction>
         <owl:onProperty rdf:resource="#playedBy"/>
         <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">
            1
         </owl:minCardinality>
      </owl:Restriction>
   </rdfs:subClassOf>
</owl:Class>

<owl:ObjectProperty rdf:ID="playedBy">
   <rdfs:domain rdf:resource="#Role"/>
   <rdfs:range rdf:resource="#Agent"/>
</owl:ObjectProperty>

<owl:Class rdf:about="#UniqueRole">
   <rdfs:subClassOf>
      <owl:Restriction>
         <owl:onProperty rdf:resource="#playedBy"/>
         <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">
            1
         </owl:cardinality>
      </owl:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
      <owl:Restriction>
         <owl:onProperty rdf:resource="#headOf"/>
         <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">
            0
         </owl:minCardinality>
      </owl:Restriction>
   </rdfs:subClassOf>
</owl:Class>

<owl:ObjectProperty rdf:ID="headOf">
   <rdf:type rdf:resource="&owl;FunctionalProperty"/>
   <rdfs:domain rdf:resource="#UniqueRole"/>
   <rdfs:range rdf:resource="#Holon"/>
</owl:ObjectProperty>

<owl:Class rdf:about="#Holon">
   <rdfs:subClassOf>
      <owl:Restriction>
         <owl:onProperty rdf:resource="#assignedTo"/>
         <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">
            1
         </owl:cardinality>
      </owl:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
      <owl:Restriction>
         <owl:onProperty rdf:resource="#consistOf"/>
         <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">
            1
```

```
         </owl:minCardinality>
      </owl:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
      <owl:Restriction>
         <owl:onProperty rdf:resource="#hasHead"/>
         <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">
            1
         </owl:minCardinality>
      </owl:Restriction>
   </rdfs:subClassOf>
</owl:Class>

<owl:FunctionalProperty rdf:ID="assignedTo">
   <rdfs:domain rdf:resource="#Holon"/>
   <rdfs:range rdf:resource="#CooperativeDomain"/>
</owl:FunctionalProperty>

<owl:ObjectProperty rdf:ID="hasHead">
   <rdf:type rdf:resource="&owl;FunctionalProperty"/>
   <owl:inverseOf rdf:resource="#headOf"/>
</owl:ObjectProperty>

<owl:Class rdf:about="#Agent">
   <rdfs:subClassOf>
      <owl:Restriction>
         <owl:onProperty rdf:resource="#plays"/>
         <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">
            1
         </owl:minCardinality>
      </owl:Restriction>
   </rdfs:subClassOf>
</owl:Class>

<owl:ObjectProperty rdf:ID="plays">
   <owl:inverseOf rdf:resource="#playedBy"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="consistOf">
   <rdfs:domain rdf:resource="&owl;Thing"/>
   <rdfs:range rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#List"/>
</owl:ObjectProperty>

<owl:TransitiveProperty rdf:ID="subDomainOf">
   <rdfs:domain rdf:resource="#CooperativeDomain"/>
   <rdfs:range rdf:resource="#CooperativeDomain"/>
</owl:TransitiveProperty>

<owl:Class rdf:about="#Job">
   <rdfs:subClassOf>
      <owl:Restriction>
         <owl:onProperty rdf:resource="#consistOf"/>
         <owl:minCardinality>2</owl:minCardinality>
      </owl:Restriction>
   </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:about="#Holon">
   <rdfs:subClassOf>
      <owl:Restriction>
         <owl:onProperty rdf:resource="#consistOf"/>
         <owl:minCardinality>1</owl:minCardinality>
      </owl:Restriction>
   </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:about="#Holarchy">
   <rdfs:subClassOf>
      <owl:Restriction>
         <owl:onProperty rdf:resource="#consistOf"/>
```

```
            <owl:minCardinality>1</owl:minCardinality>
         </owl:Restriction>
      </rdfs:subClassOf>
   </owl:Class>
</rdf:RDF>
```